DOCUMENT RESUME

ED 316 057 FL 018 381

AUTHOR Gothard, James E.
TITLE Building Composite Characters on a Postscript Printer.
INSTITUTION Illinois Univ., Urbana. Language Learning Lab.
REPORT NO LLL-T-16-89
PUB DATE Jul 89
NOTE 56p.
AVAILABLE FROM Language Learning Laboratory, University of Illinois at Urbana-Champaign, G70 Foreign Languages Building, 707 South Mathews, Urbana, Illinois 61801.
PUB TYPE Guides - Non-Classroom Use (055) -- Statistical Data (110)

EDRS PRICE MF01/PC03 Plus Postage.
DESCRIPTORS *Computer Printers; *Computer Software; *Diacritical Marking; *Media Adaptation; *Orthographic Symbols; *Printing; Programing
IDENTIFIERS Apple Macintosh; Laser Printers; Microsoft Word; *PostScript Language

ABSTRACT

Procedures enabling the placement of diacritical markings over a character for printing in PostScript fonts on an Apple LaserWriter printer are described. The procedures involve some programming in the PostScript Language and manipulation of Adobe PostScript fonts. It is assumed that Microsoft Word will be used to create the text to be printed. The Word/LaserWriter interface is described, and background work is outlined. Program design and specifications and definitions of terms are given, and specific procedures within the program are described, including accent placement, the accents dictionary, font operation, character-building, font modification, and modifications of Microsoft Word. Problems, unknown variables, and future development potential are discussed briefly. Appended materials include character width and encoding tables, a sample printout, program codes, and a sample Microsoft Word 4.0 initializing file. (MSE)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED316057

Language Learning Laboratory

LLL

TECHNICAL REPORT NO. LLL-T-16-89
JULY 1989

The School of Humanities

University of Illinois
at Urbana-Champaign

BUILDING COMPOSITE CHARACTERS ON A POSTSCRIPT PRINTER

JAMES E. GOTHARD

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☑ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Cheng, C. C.

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

BEST COPY AVAILABLE

LANGUAGE LEARNING LABORATORY
College of Liberal Arts and Sciences
University of Illinois at Urbana-Champaign

Technical Report No. LLL-T-16-89

Building Composite Characters on a PostScript Printer

James E. Gothard

Systems Planner I
Language Learning Laboratory

July 1989

Available from: Language Learning Laboratory, University of Illinois at Urbana-Champaign, G70 Foreign Languages Building, 707 S. Mathews, Urbana, Illinois 61801

Contents

Introduction 1
Description of Word and LaserWriter Interface 1
Background Work 2
Design Specifications 2
Program Design 3
Definition of Terms 4
Program Description 4
Description of Placit Program 4
Description of Accents Dictionary 6
Description of Font Operation 8
Description of BuildChar Procedures 9
Description of MixFont Program 10
Modifying the Microsoft Word 4.0 .ini file 12
Building New Composite Characters 12
Problems and Unknowns 15
Future Development Potential 16

Appendix

A. Character Width Tables (Wx) 17
B. Character Encoding Tables 18
Standard PostScript Encoding Table 18
Sample Modified Encoding Table 19
Modified Encoding Table 20
Blank Encoding Table 21
C. Sample Printout from Placit Program 22
D. Placit.ps Program Code 23
E. Accents Dictionary Code 26
F. BuildChar.ps Procedure Code 33
G. MixFont.ps Program Code 35
H. Sample Microsoft Word 4.0 Initializing File 38

Bibliography 52

Introduction

Clients of the Language Learning Laboratory extensively use the Apple LaserWriter[1] to print academic papers and publications dealing with the study of language. This requires accented characters not available in the PostScript fonts resident in the printer. Although the character and the accent are individually available in the fonts, there is no method for positioning an accent over a character.

Although our clients use both IBM and Macintosh personal computers, this problem of building accented characters arises primarily with IBM and compatible computers. The Apple Macintosh[2] operating system allows for easier handling and storage of graphics and fonts. There are more fonts available for these Mac computers and more programs for modifying those fonts. Besides, the Mac version of Microsoft[3] Word contains a provision they call "formula" which uses brief programming commands inserted in text to position an accent over a character. However, our focus necessitated the development of accented characters which print from an IBM word processor on a PostScript laser printer.

This report describes procedures which enable one to place diacritics over a character for printing on a PostScript printer. These procedures involve some programming in the PostScript Language[4] and manipulation of Adobe PostScript fonts. We had experience using Microsoft Word (Word) and as it openly interfaces with the laser printer we assume that Word will be used to create the text to be printed.

Description of Word and LaserWriter Interface

Adobe PostScript is the page description language used to drive the Apple LaserWriter printer. The printer is software driven: it interprets the PostScript procedures forwarded to it by the word processor in order to format the text on the page. These formatting procedures are developed by each word processor company as a part of their PostScript printer driver. The procedures used by Microsoft Word are first communicated to the printer by means of an Initializing File (.ini) prior to any document print file. Such an initializing file must be used by all word processors printing to a PostScript device.

In addition to defining formatting procedures, the initialization file may also perform other functions. For instance, new characters can be made available. Each character must be assigned to an ASCII code between 32 and 255 before it can be used. Making these assignments is called *encoding* the font[5]. By encoding the additional accented characters named in the font with extended ASCII codes, the initializing file can make them available to the word processor.

In the case of Microsoft Word 4.0, besides revising the font encoding and establishing formatting procedures, a new PostScript font called *LineDraw* is defined in the initialization file. LineDraw is a "user defined font" which draws horizontal and vertical lines in the text.

---

1 *Apple*, *LaserWriter*, and *LaserWriter Plus* are trademarks of Apple Computer, Inc.

2 *Macintosh* is a trademark licensed to Apple Computer, Inc.

3 *Microsoft* and *Word* are registered trademarks of Microsoft Corporation.

4 *PostScript* is a trademark of Adobe Systems Incorporated.

5 For more discussion of encoding fonts, see pages 93-95 in the *PostScript Language Reference Manual* or pages 207-214 in *PostScript Language Tutorial and Cookbook*.

Background Work

Our early attempts to build accented characters used Microsoft Word (version 3.0). The method took advantage of those formatting procedures of the initializing file which processed ASCII codes greater than 127. Normally, if a higher ASCII code was found inserted in a document, the procedure called "C" in Word's PostScript driver was used to process it. "C" looked up the character with that code from the Encoding Vector (an array associating codes and labels), retrieved the instructions for drawing the character from the Character Procedures (labeled *CharStrings*), and placed it within a string format to be processed normally. To develop a new accented character, the "C" procedure was modified to test for special ASCII codes. If some special code was present, the modified "C" positioned both a base character and an accent mark in the same space on the page, thus accenting the desired character.

This worked. An accented character printed and the word processor worked normally. It meant, however, that each new accented character required a new procedure and that the code had to be revised to allow access to that procedure. Making these changes was a tedious and time consuming process. The process needed to be improved for several reasons. First, as our clients realized that it was possible to add new accents to the LaserWriter fonts, there were more requests for additional characters, and, it was too time consuming to do this using the process just described. More crucially, when Microsoft revised their initializing procedures to speed up printing, they internalized the processing of inserted ASCII codes and bypassed the "C" procedure. Although the new Word was much faster, it was no longer possible to interject the accenting procedure.

Design Specifications

The modification of Word 4.0 required a new approach to building accented characters that was totally external to Word's PostScript printer driver. We therefore developed an alternative solution, enclosing the machinery which built the accented characters totally within an Adobe font. Word could recognize a new font easily and use the character procedures within it.[6] In an attempt to streamline future changes, the new design also provided a routine method for adding new accented characters to the font.

Specifically, the solution guidelines were:

1. The procedures for positioning the accents on the characters had to be enclosed within a normal font -- invisible to user software. Additionally, this font had to behave as expected and provide all the accented characters published for that font.

2. Commonly used characters assigned to the upper ASCII codes had to match the accepted IBM assignments, so that normal ASCII text files would print without revision. New characters could replace little used IBM graphics characters.

3. A select group of upper ASCII codes needed to be reserved for newly created characters. Once a code was assigned to a character, that assignment would not be changed in future extensions of the font. This would allow newly extended fonts to continue to print previous documents.

4. The process had to contain a facility for the rapid addition and revision of composite characters.

---

[6] Defining the characters totally within a font also has the advantage that these new fonts can probably be used with other word processors. This possibility has yet to be explored.

5. The system had to be useable by individuals with little knowledge of PostScript. It would be expected, however, that the person have enough programming experience to be comfortable working directly with program code.

Program Design

The steps to effect the solution could then be listed. First, develop a procedure to place one or two characters from an Adobe font over another character within that font. Second, develop two different utility programs: (1) a character-design program to help an individual determine the offset dimensions used to position an accent over a character, and (2) a font-modifying program which would re-encode all appropriate characters and define this new font within the initializing file sent by Microsoft Word prior to printing.

The character-design program used a straightforward process. The starting position (A) on the paper is saved, a character is printed and the final position (B) is saved. Then, the position A is restored. The X and Y positions are adjusted to properly position the accent and it is printed. Position B is restored so the printing can continue. By modifying the X and Y coordinates, the user can adjust the accent until it looks good. The resulting coordinates can then be used by the font to build the new accented character.

The font-modifying program design was more difficult. It had to build a user-defined primary font which would contain the user-developed procedures for building composite characters but no actual Adobe characters. Then, it would re-encoded a copy an Adobe font to include the regular characters plus all accented characters available from Adobe. This re-encoded font would become the supporting font drawn on by the primary to print text. The process would work like this. A normal character called by the word-processor would be taken directly from the supporting font for printing. If a new composite character were called, a procedure in the primary font would pull both the character and accent from the supporting font, position and print them.[7]

A user-defined PostScript font uses a *BuildChar* procedure to position and print the character called from the font. The BuildChar procedure in our primary font would be composed of two other procedures: *Combit* (Combine it) which would decide whether a character was regular or composite, and *Mac* (Make a character) which would use the X and Y parameters from the placement utility to position and print the accented character.

The information needed to build the primary and supporting fonts was held in a PostScript dictionary[8] called *Accents*. Within the Accents dictionary is an array listing the decimal codes associated with the Adobe composite characters (*PSCodes*), another array containing the decimal codes assigned to the new composite characters (*CompCodes*), and the labels and procedures defining the new composite characters (*CharProcs*) for each Adobe font to be modified.

---

[7] The solution suggested here is based on a program provided by Adobe Systems, Inc. called *Dharmapub1*. This program builds composite characters using the primary and supporting font model. No information as to the author or development date were provided.

[8] A "dictionary" is defined as "a structure in which are stored *key-value* pairs, where the key and the value can be any valid PostScript language object. Presenting a key (usually a *name object*) to the PostScript interpreter will cause it to perform a name lookup, and execute the resulting object found in the dictionary." Reid, Glenn C. *PostScript Language Program Design*. Addison-Wesley Publishing: Reading, MA. 1988 p 4.

All of these procedures and data structures were contained within the dictionary *MxDx* to be available when needed.

We had serious doubts about the ability of one font to use another font in this manner because our access to critical font information was limited by Adobe. Also, it was possible the procedures would require more memory than available in the printer. Happily, however, this strategy worked. The memory available within the LaserWriter Plus printer allowed the modification of all the resident fonts with no memory difficulties.

Each procedure used in the composite character process is described below in detail for anyone interested in further development. However, anyone interested in simply using this process can find, under the section *Building New Composite Characters*, the information they need to modify the Accents dictionary and build their own composite characters.

Definition of Terms

There are several fonts and programs described below. To avoid confusion, it may be best at this point to offer some terms and definitions used in the following explanation. Throughout, the terms **accented characters** and **composite characters** are used interchangeably.

The **character-design program** described is called *Placit.ps*. This utility is used to determine the best position for the desired accent over the character. The **font-modifying program** described will be referred to as *MixFont.ps*. This program builds, from an Adobe base font, the primary user-defined font and its supporting font used to print composite characters.

The **primary font** is user-defined, contains the procedures for building the composite characters and is called *ForeignFont* within the MixFont program. The name assigned the primary font after modification by MixFont.ps is constructed by adding the "-Foreign" suffix to the name of the base font (e.g. Times-Roman-Foreign)[9]. This is the same convention used by Microsoft Word. Within the body of the MixFont.ps program, ForeignFont is shortened to *ForFont* to conserve space. The **supporting font** will be referred to by the name *PlusFont*.

The MixFont.ps program draws the information to describe the characters from the **Accents dictionary**. The **MxDx dictionary** contains the procedures and information used by the ForeignFont BuildChar procedure to print the characters.

Program Description

Description of Placit.ps Procedure

Placit.ps was designed as a utility for placing accents over a character and recording the coordinate information for later use in MixFont.ps. Placit.ps is used by modifying the definitions given to variables at the beginning of the program then communicating it to the printer to check the resulting character.[10]

---

9 *Times* and *Helvetica* are registered trademarks of Allied Corporation.

10 To communicate Placit.ps as a batch file, you need a simple communication program. Set the mode command on the communications port with 9600 baud (or to agree with the printer), 8 data bits, no parity, 1 stop bit, and no echo (e.g. MODE COM1:9600,N,8,1,P,X). An editor would also be required to modify the Placit.ps parameters within the batch file.

```
/ftsz       72                                  def
/type       /Helvetica                          def
/CharName   (fulliacute)                        def
/Accent   [ (i) (\302) -.027 .225 ]             def
```

Instructions are included in the program code shown on page 23. In the box at right, variables are shown along the left side, the define operator is along the right side, and the definitions which must be modified are in the middle. *Ftsz* refers to the point size of the example printed on the page. *Type* is the name of the font to be tested. *CharName* is the name of the new accented character. *Accent* is the array which holds the information to build the new accent. This first example contains the parameters to place an acute accent over a dotted "i" thus the name "fulliacute". Using these values and communicating Placit.ps to the printer would produce the character "í" in Helvetica type one inch tall. You can review a sample page from a Placit.ps print on page 22 in Appendix C.

To help you decipher this notation, the "/" preceding the font name identifies it as a literal PostScript object label, not an executable object, and must be included. The Charname is defined as a PostScript string by placing it between parentheses. For the most part, these offsets are arrived at by trial and error. Accent is a PostScript array (enclosed in square brackets [...]) containing the items in the bottom line of the example. This array defines the new composite character. The base character is defined first as a string. That is followed by each accent and its appropriate X and Y coordinate measures. The "(\302)" is a string containing a special character notation: an octal code, identified by the "\", referring to the acute accent in the Standard PostScript Encoding Tables (see page 18). The character coordinate scale is a 1000 by 1000 unit grid so offset measurements are entered as a three-place decimal number (.123)[11]. The -.027 and .225 supply the X and Y offsets to position the accent properly. Placit.ps is currently designed to place a maximum of two accents on a single character but it could be modified to display more.

```
/ftsz       72                                  def
/type       /Times-BoldItalic                   def
/CharName   (udiergrave)                        def
/Accent
 [ (u) (\310) .111 0 (\301) .131 .190 ] def
```

In a second example, Placit.ps is used to print a character with two accents. It is called "udiergrave" as it contains the base character "u" accented by both a dieresis and a grave. The type used is Times-BoldItalic and the size of the test character remains the same. It produces the character "ǜ". The font size of 72 points, one inch, makes a character large enough that small changes in positioning can be easily seen. This size, of course, can be changed to get a different view of the character. In developing a new composite character, the different faces such as Bold, Italic, BoldItalic, Narrow must be tested because each one requires different offset measurements (e.g. A "bold" font has wider characters than a "narrow" font.).

As Placit.ps is intended to be a utility for positioning accents on characters, there is no need to discuss its PostScript code in detail. It is listed on page 23 for your information. You may find it helpful to refer to the instructions given at the beginning of the program. Also, you will notice from the output example on page 22 that the character bounding box and width figures are listed. These would be the parameters for the *setcachedevice* operator if it were used but are not required for the process we are discussing here.[12]

---

11 One of the strengths of PostScript is its ability to translate an object from one coordinate system, or grid, to another. The space used most often to describe an object in PostScript is the *user coordinate system* where a unit is arbitrarily defined as a point (approximately 1/72 of an inch) with the 0 0 point in the lower left corner. However, each character defined within the PostScript font machinery is described using the *character coordinate system* or a 1000 unit by 1000 unit grid. As the character is printed, this character coordinate system is translated by the PostScript commands (*scalefont* or *makefont*) to present the desired font size.

12 For more information on the setcachedevice procedure refer to the *PostScript Language Reference Manual* from Adobe Systems published by Addison-Wesley.

The character width coordinates used in the character procedures match ones listed in the Adobe AFM Files.[13] An abbreviated table of character widths can be found on page 17.

Description of Accents Dictionary

After the offset measurements for placing the accent in the composite character are known, the information describing the composite character is all brought together to make it easier to maintain and add new composite characters. The information is defined in a PostScript dictionary called *Accents*. A complete listing of Accents is given on page 26. It contains the following:

1. *Definitions* -- the variables *FontCount* and *CharCount* are defined as the number of fonts and new composite characters included in the dictionary.

2. *PSCodes* -- an array (beginning on line 5) listing the codes assigned composite characters designed by Adobe Systems in the supporting font.

3. *CompCodes* -- an array (beginning on page 28) listing the codes assigned the new composite characters.

4. *CharProcs* -- dictionaries (beginning on page 29) containing the character procedures and parameters in executable strings. These are matched with the names of composite characters in each font which can be modified with MixFont.ps.

The first section of the Accents dictionary defines the variables used to build the subdirectories holding the character information. fonts are added. Font Count refers to the number of fonts recognized in the Accents dictionary. CharCount refers to the number of different composite characters defined within each font. As the number of characters and fonts increase, more size is allocated within the sub-dictionary. In our example 8 fonts (the Times family and Helvetica family) and 16 composite characters are defined.[14]

```
/FontCount  8 def
/CharCount 16 def
```

PSCodes holds the codes for Adobe composites. Within each Adobe font are characters that are unencoded. "Unencoded" means the character is defined within the font but not included as a coded character available to the word processor. Within our modified font, we wanted to include as many of these Adobe accented characters as possible. PSCodes lists the codes which are used to include this full range of Adobe accented characters. All of the fonts modified by the MixFont.ps program are encoded with these additional Adobe characters. The complete list of unencoded characters can be found on page 254 in the *PostScript Language Reference Manual*. PSCodes would only need to be modified if Adobe made new composite characters available.

Take a look at the Modified Encoding Chart on page 20. The Adobe characters currently available are all included in this listing but not necessarily in the same order as those found in Adobe's standard encoding (see page 18). It was important to reserve codes for future composite characters. Thus, the Adobe characters were given numbers following those of the regular characters in the font (i.e. codes 128-201).

---

13 *AFM Files: An Interchange Format for PostScript Font Metrics*. Adobe Systems, Inc. 1984.

14 The size of the Accents dictionary is increased by 2 when it is created to accommodate the PSCodes array and the CompCodes array as well as the fonts.

The symbols and accents themselves were grouped at the end of the list (codes 225-254). Newly built composite characters can then be assigned to the remaining codes (codes 202-224) defined as */.notdef*. Also, in re-encoding the Adobe composites, they were placed to agree with conventional ASCII tables.

Moving to page 28 of Appendix E, the CompCodes section of the dictionary lists the codes assigned to new composite characters. As a new character is defined, it is assigned a number and listed in this area by name. Again, the "/" tells PostScript this is a label and not a executable procedure. Codes listed in CompCodes override those defined in PSCodes so the /.notdef previously assigned to the code does not have to be changed.

Page 29 begins the CharProcs or procedures which build the new composite characters. There is a procedure defined for each character and each font. Each sub-dictionary is identified by the name of its related font for easier access. Within each font's sub-dictionary, the name of the character is given as the key followed by its procedural string. There is space provided in the size of the dictionary for the two CharProcs common to all the fonts. These two, /.usual and /.notdef, are added during MixFont.ps' modifications (see the middle of page 36).

The box below shows two typical CharProcs, one "acaron" defining an "a" with a caron accent, the other, a "u" with both a dieresis and a macron accent. In these examples, the literal name begins with "/" and the procedural string is enclosed by "{ }" curly brackets. The format is very similar to that used with the

```
/acaron {(\141) (\365) 55 0 444 5 Mac) def
/udiermacron {(\165) (\367) 73 0 (\364) 73 160 500 8 Mac) def
```

Placit.ps utility in that the first entry defines the base character followed by the accent(s) with their respective X and Y offset coordinates. There are several differences however.

First, there are three additional entries at the end of the procedure: two numbers and the procedure call *Mac*. The first of these two numbers is the *X width (Wx)* of the character. This width is taken from the AFM Tables discussed earlier or from the Placit.ps printout (see page 22). The second number is simply the number of entries in the procedure. It tells Mac the number of accents to apply to the base character. The Mac procedure positions and prints the composite.

Second, the base character and each accent is referred to by its code in the PSCodes listing in the Accent dictionary and **not** from the Standard Encoding Tables of PostScript. The characters used in Placit.ps were called from the standard Adobe font. However, the word processor will call the character not from the standard font but from the re-encoded PlusFont. Even though the PSCodes are listed as decimal codes in the program, they must be referenced as octal numbers. Tables listing the octal codes are found in Appendix B (see page 18).

Lastly, the offset coordinates are listed as whole numbers and not decimals as in Placit.ps. The numbers are printed on the Placit.ps output as they should be used (see page 22).

Description of Font Operation

The discussion of the Accents dictionary looked at the character procedures (CharProcs) and the reference code (PSCodes and CompCodes). The discussion of Placit.ps looked at the process of printing a base character with one or two accents. Before going on to describe the MixFont program, it may help to summarize the operational steps between the keystroke in the word processor and the final printed character.

1. A key is pressed on the keyboard sending a code to the word processor.

2. The word processor positions these codes within the document.

3. The print instruction is issued to the word processor.

4. The initializing file (.ini) is sent to the laser printer.

5. The MixFont.ps procedure within the initializing file copies and renames the resident Adobe fonts to build new fonts: a supporting font with encoding for all of Adobe's composite characters using PSCodes from the Accents dictionary and a primary font which encodes all the new composite characters (CompCodes) and their procedures (CharProcs).

6. The word processor tells the printer where to place the next character on the paper then sends it the selected font and the character code.

If we were using a resident Adobe font, at this point the printer would (a) place that font dictionary on top of the dictionary stack, (b) find the name of the character in the Encoding Vector using the character code, (c) use that character name to call the Adobe procedure from the font's CharStrings, (d) build a bitmap of the character, (e) use the setcachedevice operator to store the bitmap for later use, and (f) print the character. (For more discussion, see the *PostScript Language Reference Manual*, Using Fonts, page 85).

What happens with our font is similar but contains the machinery which allows us to modify the process. Every non-Adobe font must be identified as User-defined (FontType = 3) and must contain a BuildChar procedure to tell the printer how to build the characters. Thus, after finding the name of the character (step b above), it goes to a BuildChar procedure to get its next instructions. Continuing our step by step example:

7. The printer opens the font called for by the word processor.

8. The printer finds a FontType 3 and it calls the BuildChar procedure.

9. The BuildChar procedure

   a. Calls the Combit procedure.

   b. Opens the MxDx dictionary to place the Combit procedure and the Mac procedure on top of the dictionary stack.

   c. The Combit procedure sets up the two fonts, tests whether the character is a normal Adobe character or a new composite character. Then, it either calls the Adobe character from the supporting font and prints it using the /.usual procedure (see page 36) or calls the proper CharProc and uses the Mac procedure to print the new character.

Description of BuildChar Procedures

As described earlier, each user-defined font in PostScript is required to use a BuildChar procedure to build each character. The supporting font, named PlusFont (used to actually print the characters) maintains the FontType assigned to it by Adobe so it can print using the Adobe character CharStrings. The ForeignFont as the user-defined font must operate through a BuildChar procedure. The code of the BuildChar procedure used here is listed beginning on page 33. It may be helpful to refer to it during this discussion.

The BuildChar procedure calls the Combit procedure which in turn:

1. Pushes the MxDx dictionary on the dictionary stack.

2. Receives the parameters presented to the BuildChar procedure by the font (i.e. the character code and the font dictionary).

3. Identifies the font used as the supporting font (the PlusFont) and sizes this font to match the character coordinate system as the current font in the graphics state.

4. Calls and executes the proper character procedure from either PlusFont if the character is normal, or ForeignFont if the character called is defined as a new composite.

The MxDx dictionary is called because it contains the procedures to build the new characters. The Mac procedural call, as you recall, is included in the character definition in the Accents dictionary. Pushing the MxDx dictionary to the top of the dictionary stack allows access to Mac when BuildChar attempts to execute the character procedure.

An entry not normally found in Adobe fonts, */BaseFont*, identifies the respective supporting PlusFont used by the ForeignFont. Combit uses this information to set the supporting font as the current font for printing. ForeignFont's dictionary is transferred as one of the parameters to the BuildChar procedure. Thus, the primary font dictionary is easily accessible on top of the dictionary stack and the supporting font is equally as accessible as the current font in the Graphics State stack. PlusFont is scaled to the 1000 by 1000 unit character coordinate system to match the grid used by BuildChar before it scales the character to match the point size specified by the word processor. The *save* and *restore* commands bracketing BuildChar respectively save the existing graphics state and restore it later.

With the proper font set, Combit checks ForeignFont's character definitions in the CharProcs dictionary. If the character is found, Combit retrieves the definition from the font and executes it. Execution means simply placing the parameters for building the composite character on the stack and then executing the Mac procedure to print the character. If the character called is not known in the CharProcs dictionary, Combit assumes it is a character found in PlusFont, places the character within a string and prints it. A character definition within the CharProcs dictionary called /.usual accomplishes this process. /.notdef is a default procedure to cover any unknowns and simply prints a blank space (ASCII 32). The code for these two additional procedures is shown below (and on page 36).

```
/.usual  { charStr stringwidth setcharwidth newpath
           0 0 moveto charStr show } bind put

/.notdef { charStr stringwidth setcharwidth newpath
           0 0 moveto (\040) show } bind put
```

The code in Mac is presented in Appendix F. The straightforward process:

- receives the composite character information as parameters,
- set the required charwidth procedure,
- prints the character to be accented,
- returns to the base of the grid,
- positions the accent(s), and prints.

Description of MixFont.ps Program

The program that modifies a font to include the new composite characters is called MixFont.ps. It builds two fonts: a primary font with the suffix Foreign and a supporting font (PlusFont) whose name is built from the initials of the base font with the suffix Plus (e.g. TR-Plus). The code discussed below begins on page 35.

In the first seven lines MixFont.ps defines its own dictionary and receives parameters. *CharInfo* becomes the dictionary containing the composite character definitions and codes: in our case this is Accents. *PlusFontName* holds the name the supporting font will adopt. *ForFontName* holds the name the primary font will carry. And, *BaseFontName* is the font on which the new fonts will be based. This base font name is quickly used in line 15 to call up the appropriate dictionary which is held as *BaseFontDict* for speed and convenience.

After receiving the parameters, the program sets a flag (*CharKown*) to indicate that new composite characters are actually defined for this base font. If not, the base font is re-encoded to include the full compliment of Adobe accented characters. Unfortunately, due to differences in the way the fonts are constructed, the MixFont.ps program does not work with Courier. Thus, a quick check is made to determine if the font is a Courier, and if so, the flag is set to re-encode it also by adding only the Adobe composite characters. **MixFont.ps will only re-encode fonts with the complete Adobe accented character set if they are courier or do not have new composite characters defined.**

Proceeding on, the program creates two new dictionaries: ForFontDict which will hold the entries for ForeignFont, and PlusFontDict which, as the re-encoded copy of the base font, will serve as the supporting font. MixFont.ps then copies and re-encodes the Plus. ntDict. At this point, if the product of MixFont.ps is to be only a re-encoded font (without user-defined composite characters), the program skips to the end, names the PlusFontDict as the primary font with the ForFontName, defines the dictionary as a font and prints the appropriate message to the user.

Now look at the top of page 36. If the composite characters are defined in Accents, MixFont.ps continues by naming the PlusFontDict with PlusFontName and moves to build the primary font's dictionary. It adds the name to FontName, identifies the new font as user-defined (FontType = 3), and copies the other required entries from the base font. An additional entry (/BaseFont) is defined to identify the supporting font for use in the BuildChar procedure. And, while the Encoding array is included in ForeignFont, it is not actually used so it is initialized with the standard /.notdef entry.

The new character codes are taken from the CharInfo variable and added to the Encoding array within PlusFontDict. This process was delayed until after the CharKnown decision. At this point, PlusFontDict contains all the necessary entries to be defined as the PlusFont.

The procedures that define the new composite characters are next added to ForFontDict. They are taken from CharInfo and held within a variable while the definitions common to all fonts are added. These common procedures are /.usual and /.notdef. The complete set of new composite character definitions is then defined in ForFontDict. With the addition of BuildChar, ForFontDict is ready to be defined as ForeignFont.

The following list may help clarify the relationship between ForeignFont and PlusFont.

ForeignFont	*PlusFont*
The user-defined *primary* font. It is named in the printer's FontDirectory and called by the word processor.	The Adobe *supporting* font. Used as the currentfont to print usual characters or those that make up composites.
/Encoding Not used.	**/Encoding** Contains the full set of character codes.
/CharProcs Contains the parameters and procedure call to produce a composite character.	**/CharStrings** Contains the Adobe character strings which are used to print the standard characters.
/BaseFont Names the supporting font.	**/BaseFont** Not used in an Adobe font.
/BuildChar The procedure which prints the user-defined character.	Adobe uses proprietary procedures to draw and print their characters.

Modifying the Microsoft Word 4.0 .ini File

The procedures and fonts used by Word 4.0 are defined in its initializing file. This file is sent to the PostScript printer prior to the execution of the first print command. The Accents dictionary, the MxDx dictionary containing the BuildChar procedures, and MixFont.ps must be contained within this file. They establish the new fonts and characters in the printer.

An example of a modified initializing file is presented on page 38. It is well worth your time to compare the initializing file you are now using with this one. If the initializing file you are using does not contain the added code beginning on page 42, you can modify the file by replacing the old Encoding list with the Accents dictionary (bottom of page 42 to page 48) and the re-encode program from the original file with MixFont.ps and BuildChar procedure as they appear in the appendix (see pages 49 - 51). Remove any comments and condense code to save space.

The lines at the end of the initializing file which call the Re-Encode program need to be changed to call the MixFont instead. ***It is important to make a backup of the file before you begin any modifications. Also, choose a different name for your modified file to avoid confusion.***

If you are working with a processor other than Microsoft Word, it is possible that you can still use these procedures to modify fonts and add new composite characters. If your word processor sends an initializing file similar to the one described, add the programs and Accents dictionary to that file and check your word processor's printer manual to see how to install a new font in the printer driver. If there is no initializing file, or you are programming the laser printer directly, simply build your own initializing file containing the programs to prepare the printer.

Building New Composite Characters

Here are some step by step instructions for making a new composite character and adding it to the initializing file for Microsoft Word. The steps lead you through the process of using Placit.ps to position the diacritic over or under the character, building the font procedure used to print the character, incorporating it into the font and the initializing file, and, finally, updating the character information in the Word printer driver. If you are attempting this without studying the previous information or with limited PostScript experience, you may need to refer back if you have questions.

You will need to have the following materials available to begin the process:

1. A word processor used to edit the Placit.ps program file. The file must be saved **without** formatting as a text or ASCII file.

2. A communications program to use to communicate with the Laserwriter printer through the serial port (COM1:).

3. MAKEPRD.COM used to modify the Microsoft Word printer drivers. (See the Microsoft Word Printer Manual for more information.)

4. A copy of the Placit.ps program on disk.

5. A copy of the MixFont.ps program on disk.

6. A copy of the Accents dictionary on disk.

7. A copy of the initializing file (for Word, POSTSCRP.INI)

8. A copy of the printer driver file (for Word, POSTSCRP.PRD)

Remember that the file you modify with the word processor must be saved without formatting as a text file or ASCII file before you communicate it to the printer. To add a new accented character to the font.

1. Use Placit.ps to adjust the placement of the accent over the character.

   Look at the sample Placit.ps print on page 22. In the box below are the parameter definitions inside Placit.ps used to create the sample print. As you can see, the program takes these inputs, prints a sample character for your inspection, furnishes a listing of the offset coordinates (Ax, Ay, Bx, By), the character width (Wx), and, though you should not require them, the dimensions from the lower left to the upper right corner of the bounding box which encloses the entire character.

```
/ftsz       72                                          def
/type       /Times-BoldItalic                           def
/CharName   (udiergrave)                                def
/Accent     [ (u) (\310) .111 0 (\301) .131 .190 ]      def
```

   To use Placit.ps you will need to have available the new name for the character and the octal codes from the *Standard PostScript Encoding Table* for the base character and accent(s). You can find these codes in the appendix[15] on page 18.

   Remember to enter the X and Y offsets as decimal numbers (i.e. .123). The offset measurements from the Accents dictionary procedures (see pages 29-32) may be helpful guides. Find the listing for the font you are using, if possible. If not, find a font which has similar characteristics to your font. Look for a character and accent similar to the ones you want to use and begin with the same x and y coordinates used there.

   Set the mode in your computer by typing at the prompt: "mode com1:9600,n,8,1,p,x". After you have used the word processor to modify the Placit.ps file to your specifications, forward the modified file to the laser printer using the communications program. After evaluating the printout from Placit.ps, if the character needs adjustment, adjust the offset measurements by trial and error until the character's appearance is satisfactory.

   This process must be repeated for all the fonts in which the character will be used. In our Accents dictionary we have entries for eight fonts, the complete family of Helvetica and Times-Roman.

---

15 Taken from the *PostScript Language Reference Manual*, page 252.

2. Assign the new character a code for the Accents dictionary -- a decimal number.

   If you use the Modified Encoding Table from page 20 as a guide, you will find the codes between 202 and 224 are available without interfering with common font characters.

3. Add the new character code and character name pair to the CompCodes list in the Accents dictionary.

   The new character name and code are identified within the Accents dictionary. As an example, the character code and name at right are added, in sequence, to the CompCodes array in the Accents dictionary as shown at the top of page 29.

   ```
   220  /udiergrave
   ```

4. Update the number of characters variable CharCount at the beginning of the Accents dictionary.

   Notice on page 26, that the number 16 refers to the number of characters listed in the CompCodes array (see page 28). This variable should contain the total number of characters defined for each font. All the characters must be defined for each font in Accents. If a new font is included in the dictionary, be sure to increase the variable FontCount also.

5. Determine the character's width (Wx) from the Placit.ps printout.

   For the complete information about each character consult the AFM (Adobe Font Metrics) Files from Adobe Systems. A partial width information table is presented in Appendix A on page 17 for your convenience. As a general rule, you can use the width of the base character in the definition procedure.

6. Build the character procedure for each font in Accents using the following format:

   /name {Char Accent Xoffset Yoffset Xwidth 5 Mac} def

   where "Char" is the base character octal code; "Accent" is the accent octal code assigned from the Accents directory; offsets are taken off the Placit.ps print. If two accents are placed on the character, use the following:

   /name {Char Accent1 Xoffset Yoffset Accent2 Xoffset Yoffset Xwidth 8 Mac} def

   Both the characters and accents must be identified with octal codes (\###) taken from the Accent dictionary PSCodes. In the case of our sample, you could get those codes from the Modified Encoding Chart on page 20 in the appendix so that it matches the codes in the Accents dictionary. The procedure will actually be calling for the character from the supporting font. The offset measurements are entered as an integer, not as a decimal.

The complete procedure for our udiergrave can be seen on page 30, near the bottom, or in the box below.

```
/udiergrave { (\165) (\367) 111 0 (\360) 131 190 556 8 Mac } def
```

A similar procedure must be added for each font in the Accents dictionary. Use the information from the Placit.ps printouts to build each procedure.

7. Add the character width (Wx) to the printer (.prd) file for Microsoft Word 4.0.

   Use the MakePrd.com program that accompanies Microsoft Word to translate the PostScript printer (postscrp.prd) file to a text format. Within the printer file, all the fonts used by Word are described and the tables giving the width of each character is referenced. The actual tables follow after all the fonts have been described. To update the printer file, you need to add the proper width measurement (Wx) after the new composite character's code for each font. (Make a note, while reviewing the beginning of the printer file, of which font is associated with which width table.) A expanded explanation of the .prd file format can be found in the *Printer Information for Microsoft Word* Manual.

   Once each font contains the proper width measurement for the new composite character, use the MakePrd program to translate the text back to a .prd file again. You may find it prudent to rename the printer file to avoid confusion with the original printer file. ***Always make a backup copy of the original file before beginning your work.***

Copy both the new initializing file and .prd file to your Word disk, select the printer in the Print Option menu, then do a test print to see the new character in various situations. To insert your new character into a document with Microsoft Word 4.0, hold down the Alt key and enter the upper ASCII code from the keypad. What appears on the computer screen may or may not match the printed character.

Problems and Unknowns

Courier will not work with the MixFont.ps program. MixFont.ps re-encodes the Courier fonts to include all of the Adobe characters but none of the new composite characters.

It is easy to mix up the codes in the Accents dictionary character procedure or use a decimal rather than an octal number and print the wrong character. If this happens, recheck the Modified Encoding Chart on page 20 for the proper codes.

MixFont.ps will build fonts that work only on a Apple LaserWriter Plus printer or newer (PostScript version 3.0). It will not work on the original Apple LaserWriter. Stray lines and characters appear on the page when printed on the older printers.

The setcachedevice procedure which stores the bitmap of the character in the PostScript font machinery is bypassed in this program. This is because the BuildChar procedure in a user-defined font must have either

a setcharwidth or setcachedevice operator.[16] The setcharwidth operator is the easier of the two to use as it only requires the width of the character being printed but it bypasses the bitmap cacheing machinery. To use the full setcachedevice operator would have required a complete font metrics file on all the characters: a very large file and time consuming accessing task.

I opted to take the easier route as the need to print these special characters did not seem to justify the extra effort. Unfortunately this negatively affects the operation of the resultant modified fonts. Although specific measurements have not been made, the processing of the font is significantly slower than one in which the bitmaps of the characters are saved routinely for later recall by the printer. Therefore, it is best to use the modified fonts only when necessary by printing only specific pages and use an Adobe font for routine printing not requiring any special composite characters.

This project does not address the need to build corresponding screen fonts. The screen will show the standard ASCII character unless the resident screen fonts are revised to show the new accents. Do to competition for memory by applications and the IBM character-based screen, building screen fonts for the IBM is almost impossible. As the new Windows environment develops and is more widely accepted, screen fonts for the IBM should become available.

Future Development Potential

Possible efforts to develop this process further or make it more efficient could be:

1. Devise a method for including the setcachedevice procedure in the BuildChar procedure. This would save the bitmap of the new composite character for faster access by the printer.

2. Improve the user interface so that modification does not involve the actual modification of code.

3. Provide controls on the size and number of the characters in the composite. This would allow composite characters to include other character combinations not necessarily involving accents.

4. Develop and test methods to apply this procedure and PostScript interface to other popular word processors.

---

16 For more information about the BuildChar procedure see the PostScript Reference Manual, pages 100-102. For more about the setcachedevice and setcharwidth operator see pages 212 -13 in the same manual.

Appendix

A. Character width tables (Wx).

Character	Times	Bold	Helvetica	Bold
A / a	722 / 444	722 / 500	667 / 556	722 / 556
B / b	667 / 500	667 / 556	667 / 556	722 / 611
C / c	667 / 444	722 / 444	722 / 500	722 / 556
D / d	722 / 500	722 / 556	722 / 556	722 / 611
E / e	611 / 444	667 / 444	667 / 556	667 / 556
F / f	556 / 333	611 / 333	611 / 278	611 / 333
G / g	722 / 500	778 / 500	778 / 556	778 / 611
H / h	722 / 500	778 / 556	722 / 556	722 / 611
I / i	333 / 278	389 / 278	278 / 222	278 / 278
J / j	389 / 278	500 / 333	500 / 222	556 / 278
K / k	722 / 500	778 / 556	667 / 500	722 / 556
L / l	611 / 278	667 / 278	556 / 222	611 / 278
M / m	889 / 778	944 / 833	833 / 833	833 / 889
N / n	722 / 500	722 / 556	722 / 556	722 / 611
O / o	722 / 500	778 / 500	778 / 556	778 / 611
P / p	556 / 500	611 / 556	667 / 556	667 / 611
Q / q	722 / 500	778 / 556	778 / 556	778 / 611
R / r	667 / 333	722 / 444	722 / 333	722 / 389
S / s	556 / 389	556 / 389	667 / 500	667 / 556
T / t	611 / 278	667 / 333	611 / 278	611 / 333
U / u	722 / 500	722 / 556	722 / 556	722 / 611
V / v	722 / 500	722 / 500	667 / 500	667 / 556
W / w	944 / 722	1000 / 722	944 / 722	944 / 778
X / x	722 / 500	722 / 500	667 / 500	667 / 556
Y / y	722 / 500	722 / 500	667 / 500	667 / 556
Z / z	611 / 444	667 / 444	611 / 500	611 / 500

B. Character encoding tables

Standard PostScript Encoding Chart

Dec	Octal	0	1	2	3	4	5	6	7
0	00x								
8	01x								
16	02x								
24	03x								
32	04x	space	!	"	#	$	%	&	'
40	05x	(	)	*	+	,	-	.	/
48	06x	0	1	2	3	4	5	6	7
56	07x	8	9	:	;	<	=	>	?
64	10x	@	A	B	C	D	E	F	G
72	11x	H	I	J	K	L	M	N	O
80	12x	P	Q	R	S	T	U	V	W
88	13x	X	Y	Z	[	\	]	^	_
96	14x	`	a	b	c	d	e	f	g
104	15x	h	i	j	k	l	m	n	o
112	16x	p	q	r	s	t	u	v	w
120	17x	x	y	z	{	|	}	~	
128	20x	Ç	ü	é	â	ä	à	å	ç
136	21x	ê	ë	è	ï	î	ì	Ä	Å
144	22x	É	æ	Æ	ô	ö	ò	û	ù
152	23x	ÿ	Ö	Ü	¢	£	¥		ƒ
160	24x	á	í	ó	ú	ñ	Ñ	ª	º
168	25x	¿	⌐	¬	«	‹	›	«	»
176	26x		–	†	‡	·		¶	•
184	27x	‚	„	”	»	…	‰		¿
192	30x		`	´	^	~	¯	˘	˙
200	31x	¨		°	¸		˝	˛	ˇ
208	32x	—							
216	33x								
224	34x		Æ		ª				
232	35x	Ł	Ø	Œ	º				
240	36x		æ				ı		
248	37x	ł	ø	œ	ß				

Sample Modified Encoding Chart

Dec	Octal	0	1	2	3	4	5	6	7
0	00x								
8	01x								
16	02x								
24	03x								
32	04x	space	!	"	#	$	%	&	'
40	05x	(	)	*	+	,	-	.	/
48	06x	0	1	2	3	4	5	6	7
56	07x	8	9	:	;	<	=	>	?
64	10x	@	A	B	C	D	E	F	G
72	11x	H	I	J	K	L	M	N	O
80	12x	P	Q	R	S	T	U	V	W
88	13x	X	Y	Z	[	\	]	^	_
96	14x	`	a	b	c	d	e	f	g
104	15x	h	i	j	k	l	m	n	o
112	16x	p	q	r	s	t	u	v	w
120	17x	x	y	z	{	|	}	~	
128	20x	Ç	ü	é	â	ä	à	å	ç
136	21x	ê	ë	è	ï	î	ì	Ä	Å
144	22x	É	æ	Æ	ô	ö	ò	û	ù
152	23x	ÿ	Ö	Ü	¢	£	¥		ƒ
160	24x	á	í	ó	ú	ñ	Ñ	ª	º
168	25x	¿	⌐	¬	½	¼	¡	«	»
176	26x	▯	▯	Á	Â	À	©	╣	╠
184	27x	╦	╩	╬	¤	ð	Ð	Ê	Ë
192	30x	È	ı	Í	Î	Ï	┘	┌	Ø
200	31x	Œ	œ	Ä	Ì	Ó	ß	Ô	Ò
208	32x	µ	þ	Þ	Ú	Û	Ù	ý	Ý
216	33x	¯	´		±	‗	¾	¶	§
224	34x	÷	¸	°	¨	·	¹	³	²
232	35x	—	¼	½	¾	‘	’	÷	‡
240	36x	‚	„	‰	Â	Ê	Á	Ë	È
248	37x	Í	Î	Ï	Ì	Ó	Ô	Ò	

The above chart shows a standard encoding chart which has been modified by encoding the Adobe accented characters into the font. Then sample new composite characters have been assigned to various codes between 202 and 224. This chart was printed using the initialization file beginning on page 38. The codes 202 - 224 in the Modified Chart on the next page have been left blank as a worksheet for new composite characters.

Modified Encoding Chart

Dec	Octal	0	1	2	3	4	5	6	7
0	00x								
8	01x								
16	02x								
24	03x								
32	04x	space	!	"	#	$	%	&	'
40	05x	(	)	*	+	,	-	.	/
48	06x	0	1	2	3	4	5	6	7
56	07x	8	9	:	;	<	=	>	?
64	10x	@	A	B	C	D	E	F	G
72	11x	H	I	J	K	L	M	N	O
80	12x	P	Q	R	S	T	U	V	W
88	13x	X	Y	Z	[	\	]	^	_
96	14x	`	a	b	c	d	e	f	g
104	15x	h	i	j	k	l	m	n	o
112	16x	p	q	r	s	t	u	v	w
120	17x	x	y	z	{	|	}	~	
128	20x	Ç	ü	é	â	ä	à	å	ç
136	21x	ê	ë	è	ï	î	ì	Ä	Å
144	22x	É	æ	Æ	ô	ö	ò	û	ù
152	23x	ÿ	Ö	Ü	¢	£	¥		ƒ
160	24x	á	í	ó	ú	ñ	Ñ	ª	º
168	25x	¿	⌐	¬	½	¼	¡	«	»
176	26x	ã	õ	Ø	Œ	œ	À	Ã	Õ
184	27x	¨	´	†	¶	©	®	™	ß
192	30x	◊	‰	Û	Ù	ı	Ÿ	Ž	ž
200	31x	Œ	œ						
208	32x								
216	33x								
224	34x		ß	ı	‡	¤	‚	„	"
232	35x	—	˜	˛	˝	˙	ø	†	‡
240	36x	`	´	ˆ	˜	¯	˘	˙	¨
248	37x	°	•	·	…	¶	Ł	ł	

Blank Encoding Table

Dec	Octal	0	1	2	3	4	5	6	7
0	00x								
8	01x								
16	02x								
24	03x								
32	04x								
40	05x								
48	06x								
56	07x								
64	10x								
72	11x								
80	12x								
88	13x								
96	14x								
104	15x								
112	16x								
120	17x								
128	20x								
136	21x								
144	22x								
152	23x								
160	24x								
168	25x								
176	26x								
184	27x								
192	30x								
200	31x								
208	32x								
216	33x								
224	34x								
232	35x								
240	36x								
248	37x								

C. Sample printout from *Placit.ps* program

Times-BoldItalic

	Wx: 556.0 Wy: 0.0 LLx: 47.6247 LLy: -12.96 URx: 520.9 URy: 690.22
udiergrave	Ax: 111.0 Ay: 0 Bx: 131.0 By: 190.0

D. *Placit.ps* program code

```
%%  NAME:  Accent Placement Procedure
%%  FILE:  placit ps  (short name placit)
%%  USE:   Used for placing the accents over characters within a
%%         font in order to add new accented characters.
%%
%%  AUTHOR:  Jim Gothard
%%  DATE:    August 3, 1988      Version 3.0
%%  LIMITS:  Designed to place a maximum of
%%           two accents over a single character.
%%
%%  Make changes in the placement of the accent in the "Accent" array
%%  in the parameters list below.  Use thousands of a point (e.g. .123)
%%      Adjust "type" with the desired font.
%%      Adjust "ftsz" with the desired size in points.
%%      Provide the name of the character within the "( )".
%%  The characters must be identified with octal codes ( \### )
%%  from the standard PostScript encoding tables.
%%
%%  Use the following format:
%%
%%  [ (BaseChr) (\accent1) Xoffset Yoffset (\accent2) Xoffset Yoffset ]
%%
%%  Use only the array elements needed to define
%%    the character you wish to build.
%%      Minimum array length 4.
%%      Maximum array length 7.
%%
%%  To use utility, define parameters and communicate the Placit file
%%  to the Laserwriter printer.  Adjust the coordinates as desired.

%% =========  input parameters  ================================== %%

/ftsz     72                                          def % Example font size
/type     /Times-BoldItalic                           def % Desired font
/CharName (udiergrave)                                def % Name of character
/Accent   [ (u) (\310) .111 0 (\301) .131 .190 ]      def % Character array

%% ============================================================== %%
```

```
/str {35 string cvs show} bind def                          % make str & print.
/in {72 mul} bind def                                       % points to inches.
/adj {ftsz mul} bind def                                    % adj to font size.
/cnv {1000 mul 35 string cvs show} bind def                 % make str & print.
/GetType {findfont exch scalefont setfont} bind def         % install font.
/Go {2.5 in 7 in moveto} bind def                           % position on page.
/num {1 neg} bind def                                       % define a number.
/next {/num num 1 add def} bind def                         % add one to "num".
/name (Acc_) def                                            % define a name and
/ltr 64 def                                                 % suffix for name.
/new {/ltr ltr 1 add def} bind def                          % advance one ltr.
/newname {name 3 new ltr put name cvlit} bind def           % build next name.
/more Accent length 1 sub 3 idiv 1 gt def                   % 1 or 2 accents?

Accent                                                      % get offsets
  {newname Accent next num get def}
forall
1000 type GetType                                           % get test font

AccA stringwidth 0 0 moveto AccA true charpath              % describe font
AccC AccD moveto AccB true charpath                         % bounding box
more                                                        % & stringwidth
  {AccF AccG moveto AccE true charpath}
if
flattenpath pathbbox

/URy exch def                                               % assign values
/URx exch def
/LLy exch def
/LLx exch def
/Wy exch def
/Wx exch def

newpath  ftsz type getit                                    % get font to prt
Go AccA show                                                % print character
Go AccC adj AccD adj rmoveto AccB show                      % position accent
more                                                        % print second
  {Go AccF adj AccG adj rmoveto AccE show}                  % char if needed
if

.5 setlinewidth                                             % print rest
4.5 in 9 in moveto 4.5 in 2.5 in lineto                     % of form
1.5 in 5 in moveto 7.5 in 5.0 in lineto stroke
24 /Times-Roman GetType                                     % get printing font
2.0 in 4.5 in moveto CharName show                          % print name
```

```
18 /Times-Roman GetType                    % get printing font
5.0 in 9.5 in moveto type str              % print font used
5 in 8.5 in moveto (Wx:) show              % print bounding
6 in 8.5 in moveto Wx str                  % box dimensions
5 in 8.0 in moveto (Wy:) show
6 in 8.0 in moveto Wy str
5 in 7.5 in moveto (LLx:) show
6 in 7.5 in moveto LLx str
5 in 7.0 in moveto (LLy:) show
6 in 7.0 in moveto LLy str
5 in 6.5 in moveto (URx:) show
6 in 6.5 in moveto URx str
5 in 6.0 in moveto (URy:) show
6 in 6.0 in moveto URy str

5 in 4.5 in moveto (Ax:) show              % print accent
6 in 4.5 in moveto AccC cnv                % coordinates
5 in 4.0 in moveto (Ay:) show
6 in 4.0 in moveto AccD cnv
more {                                     % print second
  5 in 3.5 in moveto (Bx:) show            % coordinates if
  6 in 3.5 in moveto AccF cnv              % required.
  5 in 3.0 in moveto (By:) show
  6 in 3.0 in moveto AccG cnv
} if
showpage                                   % print page
```

E. *Accents* dictionary program code

```
/FontCount 8 def                          %% number of fonts
/CharCount 16 def                         %% number of new chars

/Accents FontCount 2 add dict def         %% Accents defined with
Accents begin                             %% room for 2 code lists
 /PSCodes [                               %% PostScript Font Codes
  128 /Ccedilla
  129 /udieresis
  130 /eacute
  131 /acircumflex
  132 /adieresis
  133 /agrave
  134 /aring
  135 /ccedilla
  136 /ecircumflex
  137 /edieresis
  138 /egrave
  139 /idieresis
  140 /icircumflex
  141 /igrave
  142 /Adieresis
  143 /Aring
  144 /Eacute
  145 /ae
  146 /AE
  147 /ocircumflex
  148 /odieresis
  149 /ograve
  150 /ucircumflex
  151 /ugrave
  152 /ydieresis
  153 /Odieresis
  154 /Udieresis
  155 /cent
  156 /sterling
  157 /yen
  159 /florin
  160 /aacute
  161 /iacute
  162 /oacute
  163 /uacute
  164 /ntilde
  165 /Ntilde
  166 /ordfeminine
  167 /ordmasculine
  168 /questiondown
```

```
169 /atilde
170 /otilde
173 /exclamdown
174 /guillemotleft
175 /guillemotright
176 /scaron
177 /zcaron
178 /Aacute
179 /Acircumflex
180 /Agrave
181 /Atilde
182 /Ecircumflex
183 /Edieresis
184 /Egrave
185 /Iacute
186 /Icircumflex
187 /Idieresis
188 /Igrave
189 /Oacute
190 /Ocircumflex
191 /Ograve
192 /Otilde
193 /Scaron
194 /Uacute
195 /Ucircumflex
197 /Ydieresis
198 /Zcaron
199 /Oslash
200 /OE
201 /oe
202 /.notdef
203 /.notdef
204 /.notdef
205 /.notdef
206 /Ugrave
207 /.notdef
208 /.notdef
209 /.notdef
210 /.notdef
211 /.notdef
212 /.notdef
213 /.notdef
214 /.notdef
215 /.notdef
216 /.notdef
217 /.notdef
218 /.notdef
219 /.notdef
220 /.notdef
```

```
 221 /.notdef
 222 /.notdef
 223 /.notdef
 224 /.notdef
 225 /germandbls
 226 /dotlessi
 227 /section
 228 /currency
 229 /quotesinglbase
 230 /quotedblbase
 231 /quotedblright
 232 /emdash
 233 /breve
 234 /ogonek
 235 /hungarumlaut
 236 /cedilla
 237 /oslash
 238 /dagger
 239 /daggerdbl
 240 /grave
 241 /acute
 242 /circumflex
 243 /tilde
 244 /macron
 245 /caron
 246 /dotaccent
 247 /dieresis
 248 /ring
 249 /bullet
 250 /periodcentered
 251 /ellipsis
 252 /paragraph
 253 /Lslash
 254 /lslash
  ] def

/CompCodes [                              %% Composite Character Codes
 202 /acaron
 203 /fulliacute
 204 /omacron
 205 /ucaron
 207 /amacron
 208 /emacron
 209 /ecaron
 210 /ndieresis
 213 /imacron
 214 /icaron
 215 /ocaron
 216 /umacron
```

```
 217 /udiermacron
 218 /udieracute
 219 /udiercaron
 220 /udiergrave
  ] def

/Times-Roman CharCount 2 add dict def %% Composite Character Procedures
Times-Roman begin                      %% 2 added for /.usual & /.notdef
 /acaron {(\141) (\365) 55 0 444 5 Mac} def
 /fulliacute {(\151) (\361) 0 225 278 5 Mac} def
 /omacron {(\157) (\364) 80 0 500 5 Mac} def
 /ucaron {(\165) (\365) 70 0 500 5 Mac} def
 /amacron {(\141) (\364) 55 0 444 5 Mac} def
 /emacron {(\145) (\364) 73 0 444 5 Mac} def
 /ecaron {(\145) (\365) 55 0 444 5 Mac} def
 /imacron {(\342) (\364) -27 0 333 5 Mac} def
 /icaron {(\342) (\365) -27 0 278 5 Mac} def
 /ocaron {(\157) (\365) 80 0 500 5 Mac} def
 /umacron {(\165) (\364) 73 0 500 5 Mac} def
 /ndieresis {(\156) (\367) 73 0 500 5 Mac} def
 /udiermacron {(\165) (\367) 73 0 (\364) 73 160 500 8 Mac} def
 /udieracute {(\165) (\367) 73 0 (\361) 73 180 500 8 Mac} def
 /udiercaron {(\165) (\367) 73 0 (\365) 73 180 500 8 Mac} def
 /udiergrave {(\165) (\367) 73 0 (\360) 73 180 500 8 Mac} def
end % Times-Roman

/Times-Italic CharCount 2 add dict def
Times-Italic begin
 /acaron {(\141) (\365) 89 0 500 5 Mac} def
 /fulliacute {(\151) (\361) 0 225 278 5 Mac} def
 /omacron {(\157) (\364) 89 0 500 5 Mac} def
 /ucaron {(\165) (\365) 83 0 500 5 Mac} def
 /amacron {(\141) (\364) 89 0 500 5 Mac} def
 /emacron {(\145) (\364) 68 0 444 5 Mac} def
 /ecaron {(\145) (\365) 68 0 444 5 Mac} def
 /imacron {(\342) (\364) -34 0 333 5 Mac} def
 /icaron {(\342) (\365) -34 0 278 5 Mac} def
 /ocaron {(\157) (\365) 89 0 500 5 Mac} def
 /umacron {(\165) (\364) 83 0 500 5 Mac} def
 /ndieresis {(\156) (\367) 73 0 500 5 Mac} def
 /udiermacron {(\165) (\367) 73 0 (\364) 93 180 500 8 Mac} def
 /udieracute {(\165) (\367) 73 0 (\361) 93 200 500 8 Mac} def
 /udiercaron {(\165) (\367) 73 0 (\365) 93 190 500 8 Mac} def
 /udiergrave {(\165) (\367) 73 0 (\360) 83 200 500 8 Mac} def
end % Times-Italic
```

```
/Times-Bold CharCount 2 add dict def
Times-Bold begin
 /acaron {(\141) (\365) 76 0 500 5 Mac} def
 /fulliacute {(\151) (\361) -35 225 278 5 Mac} def
 /omacron {(\157) (\364) 83 0 500 5 Mac} def
 /ucaron {(\165) (\365) 104 0 556 5 Mac} def
 /amacron {(\141) (\364) 76 0 500 5 Mac} def
 /emacron {(\145) (\364) 61 0 444 5 Mac} def
 /ecaron {(\145) (\365) 61 0 444 5 Mac} def
 /imacron {(\342) (\364) -34 0 333 5 Mac} def
 /icaron {(\342) (\365) -34 0 278 5 Mac} def
 /ocaron {(\157) (\365) 83 0 500 5 Mac} def
 /umacron {(\165) (\364) 104 0 556 5 Mac} def
 /ndieresis {(\156) (\367) 104 0 556 5 Mac} def
 /udiermacron {(\165) (\367) 104 0 (\364) 104 190 556 8 Mac} def
 /udieracute {(\165) (\367) 104 0 (\361) 104 190 556 8 Mac} def
 /udiercaron {(\165) (\367) 104 0 (\365) 104 190 556 8 Mac} def
 /udiergrave {(\165) (\367) 104 0 (\360) 104 190 556 8 Mac} def
end % Times-Bold

/Times-BoldItalic CharCount 2 add dict def
Times-BoldItalic begin
 /acaron {(\141) (\365) 83 0 500 5 Mac} def
 /fulliacute {(\151) (\361) 0 225 278 5 Mac} def
 /omacron {(\157) (\364) 83 0 500 5 Mac} def
 /ucaron {(\165) (\365) 111 0 556 5 Mac} def
 /amacron {(\141) (\364) 83 0 500 5 Mac} def
 /emacron {(\145) (\364) 55 0 444 5 Mac} def
 /ecaron {(\145) (\365) 55 0 444 5 Mac} def
 /imacron {(\342) (\364) -27 0 333 5 Mac} def
 /icaron {(\342) (\365) -27 0 278 5 Mac} def
 /ocaron {(\157) (\365) 83 0 500 5 Mac} def
 /umacron {(\165) (\364) 111 0 556 5 Mac} def
 /ndieresis {(\156) (\367) 111 0 556 5 Mac} def
 /udiermacron {(\165) (\367) 111 0 (\364) 131 190 556 8 Mac} def
 /udieracute {(\165) (\367) 111 0 (\361) 131 190 556 8 Mac} def
 /udiercaron {(\165) (\367) 111 0 (\365) 131 190 556 8 Mac} def
 /udiergrave {(\165) (\367) 111 0 (\360) 131 190 556 8 Mac} def
end % Times-BoldItalic
```

```
/Helvetica CharCount 2 add dict def
Helvetica begin
 /acaron {(\141) (\365) 111 0 556 5 Mac} def
 /fulliacute {(\151) (\361) -27 225 278 5 Mac} def
 /omacron {(\157) (\364) 111 0 556 5 Mac} def
 /ucaron {(\165) (\365) 111 0 556 5 Mac} def
 /amacron {(\141) (\364) 111 0 556 5 Mac} def
 /emacron {(\145) (\364) 117 0 556 5 Mac} def
 /ecaron {(\145) (\365) 111 0 556 5 Mac} def
 /imacron {(\342) (\364) -27 0 333 5 Mac} def
 /icaron {(\342) (\365) -27 0 278 5 Mac} def
 /ocaron {(\157) (\365) 111 0 556 5 Mac} def
 /umacron {(\165) (\364) 111 0 556 5 Mac} def
 /ndieresis {(\156) (\367) 111 0 556 5 Mac} def
 /udiermacron {(\165) (\367) 111 0 (\364) 111 165 556 8 Mac} def
 /udieracute {(\165) (\367) 111 0 (\361) 111 185 556 8 Mac} def
 /udiercaron {(\165) (\367) 111 0 (\365) 111 185 556 8 Mac} def
 /udiergrave {(\165) (\367) 111 0 (\360) 111 185 556 8 Mac} def
end % Helvetica

/Helvetica-Oblique CharCount 2 add dict def
Helvetica-Oblique begin
 /acaron {(\141) (\365) 111 0 556 5 Mac} def
 /fulliacute {(\151) (\361) -27 225 278 5 Mac} def
 /omacron {(\157) (\364) 111 0 556 5 Mac} def
 /ucaron {(\165) (\365) 111 0 556 5 Mac} def
 /amacron {(\141) (\364) 111 0 556 5 Mac} def
 /emacron {(\145) (\364) 111 0 556 5 Mac} def
 /ecaron {(\145) (\365) 111 0 556 5 Mac} def
 /imacron {(\342) (\364) -27 0 333 5 Mac} def
 /icaron {(\342) (\365) -27 0 278 5 Mac} def
 /ocaron {(\157) (\365) 139 0 556 5 Mac} def
 /umacron {(\165) (\364) 111 0 556 5 Mac} def
 /ndieresis {(\156) (\367) 111 0 556 5 Mac} def
 /udiermacron {(\165) (\367) 111 0 (\364) 131 180 556 8 Mac} def
 /udieracute {(\165) (\367) 111 0 (\361) 131 180 556 8 Mac} def
 /udiercaron {(\165) (\367) 111 0 (\365) 131 180 556 8 Mac} def
 /udiergrave {(\165) (\367) 111 0 (\360) 131 180 556 8 Mac} def
end % Helvetica-Oblique
```

```
 /Helvetica-Bold CharCount 2 add dict def
 Helvetica-Bold begin
  /acaron {(\141) (\365) 111 0 556 5 Mac} def
  /fulliacute {(\151) (\361) -27 225 278 5 Mac} def
  /omacron {(\157) (\364) 139 0 611 5 Mac} def
  /ucaron {(\165) (\365) 139 0 611 5 Mac} def
  /amacron {(\141) (\364) 111 0 556 5 Mac} def
  /emacron {(\145) (\364) 111 0 556 5 Mac} def
  /ecaron {(\145) (\365) 111 0 556 5 Mac} def
  /imacron {(\342) (\364) -27 0 333 5 Mac} def
  /icaron {(\342) (\365) -27 0 278 5 Mac} def
  /ocaron {(\157) (\365) 139 0 611 5 Mac} def
  /umacron {(\165) (\364) 139 0 611 5 Mac} def
  /ndieresis {(\156) (\367) 139 0 611 5 Mac} def
  /udiermacron {(\165) (\367) 139 0 (\364) 139 180 611 8 Mac} def
  /udieracute {(\165) (\367) 139 0 (\361) 139 200 611 8 Mac} def
  /udiercaron {(\165) (\367) 139 0 (\365) 139 190 611 8 Mac} def
  /udiergrave {(\165) (\367) 139 0 (\360) 139 200 611 8 Mac} def
 end % Helvetica-Bold

 /Helvetica-BoldOblique CharCount 2 add dict def
 Helvetica-BoldOblique begin
  /acaron {(\141) (\365) 111 0 556 5 Mac} def
  /fulliacute {(\151) (\361) -27 225 278 5 Mac} def
  /omacron {(\157) (\364) 139 0 611 5 Mac} def
  /ucaron {(\165) (\365) 139 0 611 5 Mac} def
  /amacron {(\141) (\364) 111 0 556 5 Mac} def
  /emacron {(\145) (\364) 111 0 556 5 Mac} def
  /ecaron {(\145) (\365) 111 0 556 5 Mac} def
  /imacron {(\342) (\364) -27 0 333 5 Mac} def
  /icaron {(\342) (\365) -27 0 278 5 Mac} def
  /ocaron {(\157) (\365) 139 0 611 5 Mac} def
  /umacron {(\165) (\364) 139 0 611 5 Mac} def
  /ndieresis {(\156) (\367) 139 0 611 5 Mac} def
  /udiermacron {(\165) (\367) 139 0 (\364) 159 180 611 8 Mac} def
  /udieracute {(\165) (\367) 139 0 (\361) 159 190 611 8 Mac} def
  /udiercaron {(\165) (\367) 139 0 (\365) 159 180 611 8 Mac} def
  /udiergrave {(\165) (\367) 139 0 (\360) 159 180 611 8 Mac} def
 end % Helvetica-BoldOblique
end % Accents
```

F. BuildChar.ps procedure Code

```
/MxDx 18 dict def                    % Define the workspace dict for the
MxDx begin                           %   BuildChar procedure and define
 /charStr (.) def                    %   a string variable in it.

 /Mac {                          % Define Make Composite Character Procedure
  /reps exch def                     % Define input parameters
  /Wx exch def
  /A1y exch def
  /A1x exch def
  /A1 exch def
  /more reps 5 gt def                % Flag true if more than one accent
  more                               % Define second accent if present
    { /A2y exch def
      /A2x exch def
      /A2 exch def }
  if
  /BaseChr exch def
  Wx 0 setcharwidth                  % Set width for BuildChar requirement
  0 0 moveto BaseChr show            % Print Base character
  A1x A1y moveto A1 show             % Position and print first accent
  more                               % Position and print second accent
    { A2x A2y moveto A2 show }       %   if present
  if
 } bind def

/Combit {                       % Define Combine Characters Procedure
 MxDx begin                          % Push MxDx dictionary on dict stack
  /char exch def                     % Define parameters presented to the
  /font exch def                     %   BuildChar procedure
  font /BaseFont get findfont        % Get the name of the supporting font
  1000 scalefont setfont             % Define the support font on 1000 grid
  /chrname                           % Get character name using the
    currentfont /Encoding get        %   supporting font's encoding vectors
    char get                         %   and define variable "chrname".
 def
 font /CharProcs get chrname known  % If character a new composite
  { font /CharProcs get              % get the procedures dictionary
    chrname get                      % get the procedure for that character
    newpath exec}                    % clear the graphic path and print
```

```
    { charStr 0 char put                % else put character code in a string
      font /CharProcs get               % get the procedures dictionary
      dup /.usual known                 % use the usual character if possible
        { /.usual }                     % else use the default so the program
        { /.notdef }                    %   will continue.
      ifelse
      get newpath exec }                % get the procedure and execute.
   ifelse
  end % MxDx                            % Close the MxDx dictionary workspace.
  } bind def                            % Define the Combit procedure.
end % MxDx                              % Close dict after procedures defined.
```

G. MixFont.ps program code

```
/MixFont {                                    % Define the Mix Font procedure
 /MixFontDict 25 dict def                     % Define dictionary workspace.
 MixFontDict begin                            % and push to top of dict stack
  /CharInfo exch def                          % get the four parameters:
  /PlusFontName exch def                      %  the defining dict, the
  /ForFontName exch def                       %  supporting font name, a new
  /BaseFontName exch def                      %  font name and the base font

  /CharKnown CharInfo BaseFontName known def     % flag true if new chars
  BaseFontName /Courier eq                       %  are defined.
  BaseFontName /Courier-Bold eq and              % check for Courier and
  BaseFontName /Courier-Oblique eq and           % set flag false if true
  BaseFontName /Courier-BoldOblique eq and       % COURIER WILL NOT WORK
     { /CharKnown false def }                    % WITH THIS PROCEDURE.
  if

  /BaseFontDict BaseFontName findfont def        % get base font dict.

  /ForFontDict 15 dict def                       % create both new dicts.
  /PlusFontDict BaseFontDict maxlength dict def  % PlusFont to match base

  BaseFontDict                                   % copy font to Plus dict
   { exch dup /FID ne                            % if object is not /FID
      { dup /Encoding eq                         % if object is the array
         { exch dup length array copy            % copy Encoding array
           PlusFontDict 3 1 roll put }           % to the supporting font
         { exch PlusFontDict 3 1 roll put }      % else just copy object.
        ifelse }
      { pop pop }                                % pop the /FID or Font
     ifelse                                      % ID from the stack.
  } forall

  /OldCodes CharInfo /PSCodes get def            % Add PostScript codes
  OldCodes aload pop                             % to the stack and put
  OldCodes length 2 idiv                         % in the support font.
   { PlusFontDict /Encoding get 3 1 roll put }
  repeat
```

```
CharKnown                          % IF NEW COMPOSITE CHARACTERS ARE DEFINED
 { PlusFontDict /FontName PlusFontName put          % add PlusFont name.

   ForFontDict                                      % add to ForFont
    dup /FontName ForFontName put                   % name it
    dup /FontType 3 put                             % identify as type 3
    dup /BaseFont PlusFontName put                  % identify BaseFont
    dup /FontMatrix dup BaseFontDict exch get put   % copy Matrix
    dup /FontBBox dup BaseFontDict exch get put     % copy FontBBox.
        /Encoding 256 array put                     % enter Encoding
   0 1 255                                          % initialize the
     { ForFontDict /Encoding get                    % new encoding
       exch /.notdef put }                          % array with
   for                                              % /.notdef.

   /NewCodes CharInfo /CompCodes get def            % Add new composite
   NewCodes aload pop                               % character codes
   NewCodes length 2 idiv                           % to PlusFontDict.
     { PlusFontDict /Encoding get 3 1 roll put }
   repeat

   /NewProcs CharInfo BaseFontName get def          % get new accent
   NewProcs dup                                     % char procedures
     /.notdef { (\040) stringwidth                  % add default
                setcharwidth newpath                % procedure.
                0 0 moveto (\040) show } bind put
     /.usual { charStr stringwidth                  % add the process
               setcharwidth newpath                 % to print the usual
               0 0 moveto charStr show } bind put   % characters and
   ForFontDict /CharProcs NewProcs put              % put in ForFont

   ForFontDict
   /BuildChar { MxDx /Combit get exec } put         % define BuildChar

   FontDirectory PlusFontName known not             % If not defined,
     { PlusFontName PlusFontDict                    % define the
       definefont pop }                             % new fonts
   if
   FontDirectory ForFontName known not
     { ForFontName ForFontDict definefont pop }
   if }
```

```
    { (%%[  Font Re-Encoded - New Characters Unknown  ]%%\n) print flush
      PlusFontDict /FontName ForFontName put          % if new composites
      FontDirectory ForFontName known not             % not defined
        { ForFontName PlusFontDict                    % in Accents
          definefont pop }                            % just re-encode
      if }                                            % the Base Font.
    ifelse
  end  % MixFontDict
} bind def


%% ================  TEST OUTPUT  ================================ %%

/Times-Roman /Times-Roman-Foreign /TR-Plus Accents MixFont
/Helvetica /Helvetica-Foreign /H-Plus Accents MixFont

/inch { 72 mul } def

/Times-Roman-Foreign findfont 36 scalefont setfont
2 inch 8 inch moveto
(New Accented Characters) show

/Times-Roman-Foreign findfont 18 scalefont setfont
2 inch 6.5 inch moveto
(\312   \313   \314   \315   \317) show
2 inch 5.5 inch moveto
(\320   \321   \325   \326   \327) show
2 inch 4.5 inch moveto
(\330   \331   \332   \333   \334) show
showpage

/Helvetica-Foreign findfont 36 scalefont setfont
2 inch 8 inch moveto
(New Accented Characters) show

/Helvetica-Foreign findfont 18 scalefont setfont
2 inch 6.5 inch moveto
(\312   \313   \314   \315   \317) show
2 inch 5.5 inch moveto
(\320   \321   \325   \326   \327) show
2 inch 4.5 inch moveto
(\330   \331   \332   \333   \334) show
showpage
```

H. Sample Microsoft Word 4.0 initializing file.

```
userdict /msdict known {stop} if                    % stop if already defined
{\n} print flush
serverdict begin 0 exitserver                       % exit server loop
/msdict 120 dict def                                % define dictionaries
/mslinedraw 25 dict def
/PS1 {msdict begin /ptop 792 def /sw 0 def          % describe page and format
 /ftsz 12 def /smode 0 def /STh -3.6 def /fs 0 def /offset 0 def
 fonttable 1 get fontset E 90 rotate 0 -612 translate save
 statusdict begin /waittimeout 360 def end} def
/PSp {msdict begin /ptop 792 def /sw 0 def /ftsz 12 def
 /smode 0 def /STh -3.6 def /fs 0 def /offset 0 def fonttable 1 get
 fontset E save statusdict begin /waittimeout 360 def end} def

mslinedraw begin                                    % define linedraw font
 /FontType 3 def
 /FontMatrix [1.0 30.0 div 0 0 1.0 30.0 div 0 0] def
 /FontBBox [0 0 18 30] def
 /Encoding 256 array def
 0 1 255 {Encoding exch /.notdef put} for
 Encoding 45 [
  /sh   /sh   /sh /light /med /dark
  /sv   /srj  /sdjr /dsjr /sdtr /dstr /djr  /dv
  /dtr  /dbr  /sdbr /dsbr /str  /sbl  /sjb  /sjt
  /sjl  /sh   /sjc  /sdjl /dsjl /dbl  /dtl  /djb
  /djt  /djl  /dh   /djc  /dsjb /sdjb /dsjt /sdjt
  /sdbl /dsbl /dstl /sdtl /sdjc /dsjc /sbr  /stl
  /black /bb /bl /br /bt] putinterval
 /a {moveto lineto stroke} bind def
 /b {moveto lineto lineto stroke} bind def
 /c {moveto 9 7 lineto stroke} bind def
 /d {moveto lineto lineto} bind def
 /e {lineto closepath fill} bind def
 /f {7 30 7 9 0 9 b} bind def
 /g {12 30 12 9 18 9 b} bind def
 /h {7 0 7 5 0 5 b} bind def
 /i {12 0 12 5 18 5 b} bind def
 /j {0 7 18 7 a} bind def
 /k {9 30 9 0 a} bind def
 /l {0 5 18 5 a 0 9 18 9 a} bind def
 /m {7 30 7 0 a 12 30 12 0 a} bind def
 /n {0 0 18 0 18 30 d 0 30 e} bind def
```

```
/CharProcs 55 dict def
CharProcs begin
 /.notdef {} def
 /sv {k} def
 /srj {k 0 7 c} bind def
 /sdjr {k 0 5 9 5 a 0 9 9 9 a} bind def
 /dsjr {m 0 7 7 7 a} bind def
 /sdtr {0 7 12 7 12 0 b 7 7 7 0 a} bind def
 /dstr {0 9 9 9 9 0 b 0 5 9 5 a} bind def
 /djr {f h 12 0 12 30 a} bind def
 /dv {m} def
 /dtr {h 0 9 12 9 12 0 b} bind def
 /dbr {f  0 5 12 5 12 30 b} bind def
 /sdbr {0 7 12 7 12 30 b 7 7 7 30 a} bind def
 /dsbr {0 5 9 5 9 30 b 0 9 9 9 a} bind def
 /str {0 7 9 7 9 0 b} bind def
 /sbl {9 30 9 7 18 7 b} bind def
 /sjb {j 9 30 c} bind def
 /sjt {j 9 0 c} bind def
 /sjl {k 18 7 c} bind def
 /sh {j} bind def
 /sjc {j k} bind def
 /sdjl {k 9 9 18 9 a 9 5 18 5 a} bind def
 /dsjl {m 12 7 18 7 a} bind def
 /dbl {g 7 30 7 5 18 5 b} bind def
 /dtl {i 7 0 7 9 18 9 b} bind def
 /djb {f g 0 5 18 5 a} bind def
 /djt {h i 0 9 18 9 a} bind def
 /djl {g i 7 0 7 30 a} bind def
 /dh {l} bind def
 /djc {f g h i} bind def
 /dsjb {l 9 30 9 9 a} bind def
 /sdjb {j 7 7 7 30 a 12 7 12 30 a} bind def
 /dsjt {l 9 0 9 5 a} bind def
 /sdjt {j 7 7 7 0 a 12 7 12 0 a} bind def
 /sdbl {7 30 7 7 18 7 b 12 30 12 7 a} bind def
 /dsbl {9 30 9 5 18 5 b 9 9 18 9 a} bind def
 /dstl {9 0 9 9 18 9 b 9 5 18 5 a} bind def
 /sdtl {7 0 7 7 18 7 b 12 0 12 7 a} bind def
 /sdjc {m j} bind def
 /dsjc {l k} bind def
 /sbr {0 7 9 7 9 30 b} bind def
 /stl {9 0 9 7 18 7 b} bind def
 /light {0.75 setgray n} bind def
 /med {0.50 setgray n} bind def
 /dark {0.30 setgray n} bind def
 /black {n} bind def
 /bt {0 7 18 7 18 30 d 0 30 e} bind def
 /bb {0 0 18 0 18 7 d 0 7 e} bind def
```

```
  /bl {0 0 9 0 9 30 d 0 30 e} bind def
  /br {9 0 18 0 18 30 d 9 30 e} bind def
 end %CharProcs
 /BuildChar {
  fs 1 and 0 eq {1} {3} ifelse setlinewidth
  18 0 setcharwidth
  exch begin
   Encoding exch get
   CharProcs exch get
   exec
  end
 } bind def
end %mslinedraw
FontDirectory /LineDraw known not {/LineDraw mslinedraw definefont pop}
       if

msdict begin                          % define procedures for Word 4.0
/fonttable
 [0 [/Courier-Foreign /Courier-Bold-Foreign /Courier-Oblique-Foreign
       /Courier-BoldOblique-Foreign]
  8 [/Helvetica-Foreign /Helvetica-Bold-Foreign /Helvetica-Oblique-
       Foreign /Helvetica-BoldOblique-Foreign]
  9 [/AvantGarde-Book-Foreign /AvantGarde-Demi-Foreign /AvantGarde-
       BookOblique-Foreign /AvantGarde-DemiOblique-Foreign]
  10 [/Helvetica-Narrow-Foreign /Helvetica-Narrow-Bold-Foreign
       /Helvetica-Narrow-Oblique-Foreign /Helvetica-Narrow-BoldOblique-
       Foreign]
  16 [/Bookman-Light-Foreign /Bookman-Demi-Foreign /Bookman-LightItalic-
       Foreign /Bookman-DemiItalic-Foreign]
  24 [/Times-Roman-Foreign /Times-Bold-Foreign /Times-Italic-Foreign
       /Times-BoldItalic-Foreign]
  25 [/NewCenturySchlbk-Roman-Foreign /NewCenturySchlbk-Bold-Foreign
       /NewCenturySchlbk-Italic-Foreign /NewCenturySchlbk-BoldItalic-
       Foreign]
  26 [/Palatino-Roman-Foreign /Palatino-Bold-Foreign /Palatino-Italic-
       Foreign /Palatino-BoldItalic-Foreign]
  50 [/ZapfChancery-MediumItalic-Foreign /ZapfChancery-MediumItalic-
       Foreign /ZapfChancery-MediumItalic-Foreign /ZapfChancery-
       MediumItalic-Foreign]
  56 [/Symbol /Symbol /Symbol /Symbol]
  57 [/LineDraw /LineDraw /LineDraw /LineDraw]
  60 [/ZapfDingbats /ZapfDingbats /ZapfDingbats
      /ZapfDingbats]
 ] def
/SLL {/LL def} def
/BD {STh ssc} def
/BU {STh neg ssc} def
/BN {0 ssy neg rmoveto} def
/P {exch msu exch msu neg ptop add moveto} bind def
```

```
/S {currentpoint 3 -1 roll sw 0 32 4 -1 roll widthshow
 smode 1 and 0 ne {false 1.4 ul} if
 smode 2 and 0 ne {true 1.4 ul} if
 smode 4 and 0 ne {false STh ul} if
 smode 8 and 0 ne {ldot} if pop pop} bind def
/J {msu ( ) stringwidth pop sub /sw exch def} bind def
/PE {showpage ptop exch restore save exch /ptop exch def} bind def
/SFL {msu /ptop exch def} def
/B {fs 2 and 0 eq {Bft} {BIft} ifelse
 ftsz scalefont setfont fs 1 or /fs exch def} bind def
/E {Eft ftsz scalefont setfont /fs 0 def} bind def
/I {fs 1 and 0 eq {Ift} {BIft} ifelse
  ftsz scalefont setfont fs 2 or /fs exch def} bind def
/BI {BIft ftsz scalefont setfont fs
 2 or 1 or /fs exch def} bind def
/LU {1 ssm} def
/LD {2 ssm} def
/LS {4 ssm} def
/LL {/ldmode exch def 8 ssm} def
/LE {not smode and /smode exch def} def
/F {/ftsz exch def (-) stringwidth pop 2 div neg /STh exch def
 /ftcd exch def 0 2 fonttable length 1 sub
  {dup fonttable exch get ftcd eq
   {1 add fonttable exch get fontset exit}{pop}ifelse
  }for
 E} bind def
/fontset {dup 0 get findfont /Eft exch def
          dup 1 get findfont /Bft exch def
          dup 2 get findfont /Ift exch def
          3 get findfont /BIft exch def
         } bind def
/msu {20 div} bind def
/ssc {dup /ssy exch def 0 exch rmoveto} bind def
/ssm {dup smode or /smode exch def} def
/Cf {/Eft /Courier findfont def /Bft /Courier-Bold findfont def
 /Ift /Courier-Oblique findfont def
 /BIft /Courier-BoldOblique findfont def} bind def
/ul {gsave /y exch def /dbl exch def 2 copy .2 setlinewidth
 currentpoint dbl {4 copy} if y sub newpath moveto y sub lineto stroke
 dbl {2.4 sub newpath moveto 2.4 sub lineto stroke} if grestore} bind
       def
/PSe {restore end currentfile closefile} bind def
/ldot {gsave 2 copy currentpoint /y exch def /x exch def
 newpath
 ldmode 0 eq {/offset 0 def 1 setlinewidth [1 7] 4 setdash}
 {ldmode 1 eq {[6 2] 6 setdash /offset STh def .2 setlinewidth}
 {ldmode 2 eq {[] 0 setdash /offset 1.4 def .2 setlinewidth} if}
 ifelse} ifelse
 offset sub moveto x y offset sub lineto stroke grestore} bind def
```

```
%%---------------- BEGIN ACCENTS INSTALLATION CODE -------------%%
(\n%%[  Begin MixFont procedures                          ]%%\n) print flush
(%%[    MxDx Dictionary                                  ]%%\n) print flush
/MxDx 18 dict def
MxDx begin
 /charStr (.) def
 /Mac {
  /reps exch def /more reps 5 gt def /Wx exch def
  /A1y exch def /A1x exch def /A1 exch def
  more {/A2y exch def /A2x exch def /A2 exch def} if
  /BaseChr exch def
  Wx 0 setcharwidth
  0 0 moveto BaseChr show
  A1x A1y moveto A1 show
  more {A2x A2y moveto A2 show} if
 } bind def
/Combit {
 MxDx begin
  /char exch def
  /font exch def
  font /BaseFont get findfont 1000 scalefont setfont
  /chrname currentfont /Encoding get char get def
  font /CharProcs get chrname known
   {font /CharProcs get chrname get newpath exec}
   {charStr 0 char put
    font /CharProcs get dup /.usual known {/.usual} {/.notdef} ifelse
    get newpath exec}
  ifelse
 end % MxDx
 } bind def
 (%%[    MxDx Dictionary defined                          ]%%\n) print flush
end % MxDx

(%%[    Describing Composite Characters                  ]%%\n) print flush
/FontCount 8 def
/CharCount 16 def
/Accents FontCount 2 add dict def
Accents begin
 /PSCodes [              % PostScript Font Codes
  128 /Ccedilla
  129 /udieresis
  130 /eacute
  131 /acircumflex
  132 /adieresis
  133 /agrave
  134 /aring
  135 /ccedilla
  136 /ecircumflex
```

```
137 /edieresis
138 /egrave
139 /idieresis
140 /icircumflex
141 /igrave
142 /Adieresis
143 /Aring
144 /Eacute
145 /ae
146 /AE
147 /ocircumflex
148 /odieresis
149 /ograve
150 /ucircumflex
151 /ugrave
152 /ydieresis
153 /Odieresis
154 /Udieresis
155 /cent
156 /sterling
157 /yen
159 /florin
160 /aacute
161 /iacute
162 /oacute
163 /uacute
164 /ntilde
165 /Ntilde
166 /ordfeminine
167 /ordmasculine
168 /questiondown
169 /atilde
170 /otilde
173 /exclamdown
174 /guillemotleft
175 /guillemotright
176 /scaron
177 /zcaron
178 /Aacute
179 /Acircumflex
180 /Agrave
181 /Atilde
182 /Ecircumflex
183 /Edieresis
184 /Egrave
185 /Iacute
186 /Icircumflex
187 /Idieresis
188 /Igrave
```

```
189 /Oacute
190 /Ocircumflex
191 /Ograve
192 /Otilde
193 /Scaron
194 /Uacute
195 /Ucircumflex
197 /Ydieresis
198 /Zcaron
199 /Oslash
200 /OE
201 /oe
202 /.notdef
203 /.notdef
204 /.notdef
205 /.notdef
206 /Ugrave
207 /.notdef
208 /.notdef
209 /.notdef
210 /.notdef
211 /.notdef
212 /.notdef
213 /.notdef
214 /.notdef
215 /.notdef
216 /.notdef
217 /.notdef
218 /.notdef
219 /.notdef
220 /.notdef
221 /.notdef
222 /.notdef
223 /.notdef
224 /.notdef
225 /germandbls
226 /dotlessi
227 /section
228 /currency
229 /quotesinglbase
230 /quotedblbase
231 /quotedblright
232 /emdash
233 /breve
234 /ogonek
235 /hungarumlaut
236 /cedilla
237 /oslash
238 /dagger
```

```
 239 /daggerdbl
 240 /grave
 241 /acute
 242 /circumflex
 243 /tilde
 244 /macron
 245 /caron
 246 /dotaccent
 247 /dieresis
 248 /ring
 249 /bullet
 250 /periodcentered
 251 /ellipsis
 252 /paragraph
 253 /Lslash
 254 /lslash
  ] def

/CompCodes [                    % Composite Character Codes
 202 /acaron
 203 /fulliacute
 204 /omacron
 205 /ucaron
 207 /amacron
 208 /emacron
 209 /ecaron
 210 /ndieresis
 213 /imacron
 214 /icaron
 215 /ocaron
 216 /umacron
 217 /udiermacron
 218 /udieracute
 219 /udiercaron
 220 /udiergrave
  ] def

/Times-Roman CharCount 2 add dict def
Times-Roman begin
 /acaron {(\141) (\365) 55 0 444 5 Mac} def
 /fulliacute {(\151) (\361) 0 225 278 5 Mac} def
 /omacron {(\157) (\364) 80 0 500 5 Mac} def
 /ucaron {(\165) (\365) 70 0 500 5 Mac} def
 /amacron {(\141) (\364) 55 0 444 5 Mac} def
 /emacron {(\145) (\364) 73 0 444 5 Mac} def
 /ecaron {(\145) (\365) 55 0 444 5 Mac} def
 /imacron {(\342) (\364) -27 0 333 5 Mac} def
 /icaron {(\342) (\365) -27 0 278 5 Mac} def
 /ocaron {(\157) (\365) 80 0 500 5 Mac} def
```

```
 /umacron {(\165) (\364) 73 0 500 5 Mac} def
 /ndieresis {(\156) (\367) 73 0 500 5 Mac} def
 /udiermacron {(\165) (\367) 73 0 (\364) 73 160 500 8 Mac} def
 /udieracute {(\165) (\367) 73 0 (\361) 73 180 500 8 Mac} def
 /udiercaron {(\165) (\367) 73 0 (\365) 73 180 500 8 Mac} def
 /udiergrave {(\165) (\367) 73 0 (\360) 73 180 500 8 Mac} def
end % Times-Roman

/Times-Italic CharCount 2 add dict def
Times-Italic begin
 /acaron {(\141) (\365) 89 0 500 5 Mac} def
 /fulliacute {(\151) (\361) 0 225 278 5 Mac} def
 /omacron {(\157) (\364) 89 0 500 5 Mac} def
 /ucaron {(\165) (\365) 83 0 500 5 Mac} def
 /amacron {(\141) (\364) 89 0 500 5 Mac} def
 /emacron {(\145) (\364) 68 0 444 5 Mac} def
 /ecaron {(\145) (\365) 68 0 444 5 Mac} def
 /imacron {(\342) (\364) -34 0 333 5 Mac} def
 /icaron {(\342) (\365) -34 0 278 5 Mac} def
 /ocaron {(\157) (\365) 89 0 500 5 Mac} def
 /umacron {(\165) (\364) 83 0 500 5 Mac} def
 /ndieresis {(\156) (\367) 73 0 500 5 Mac} def
 /udiermacron {(\165) (\367) 73 0 (\364) 93 180 500 8 Mac} def
 /udieracute {(\165) (\367) 73 0 (\361) 93 200 500 8 Mac} def
 /udiercaron {(\165) (\367) 73 0 (\365) 93 190 500 8 Mac} def
 /udiergrave {(\165) (\367) 73 0 (\360) 83 200 500 8 Mac} def
end % Times-Italic

/Times-Bold CharCount 2 add dict def
Times-Bold begin
 /acaron {(\141) (\365) 76 0 500 5 Mac} def
 /fulliacute {(\151) (\361) -35 225 278 5 Mac} def
 /omacron {(\157) (\364) 83 0 500 5 Mac} def
 /ucaron {(\165) (\365) 104 0 556 5 Mac} def
 /amacron {(\141) (\364) 76 0 500 5 Mac} def
 /emacron {(\145) (\364) 61 0 444 5 Mac} def
 /ecaron {(\145) (\365) 61 0 444 5 Mac} def
 /imacron {(\342) (\364) -34 0 333 5 Mac} def
 /icaron {(\342) (\365) -34 0 278 5 Mac} def
 /ocaron {(\157) (\365) 83 0 500 5 Mac} def
 /umacron {(\165) (\364) 104 0 556 5 Mac} def
 /ndieresis {(\156) (\367) 104 0 556 5 Mac} def
 /udiermacron {(\165) (\367) 104 0 (\364) 104 190 556 8 Mac} def
 /udieracute {(\165) (\367) 104 0 (\361) 104 190 556 8 Mac} def
 /udiercaron {(\165) (\367) 104 0 (\365) 104 190 556 8 Mac} def
 /udiergrave {(\165) (\367) 104 0 (\360) 104 190 556 8 Mac} def
end % Times-Bold

/Times-BoldItalic CharCount 2 add dict def
```

```
Times-BoldItalic begin
 /acaron {(\141) (\365) 83 0 500 5 Mac} def
 /fulliacute {(\151) (\361) 0 225 278 5 Mac} def
 /omacron {(\157) (\364) 83 0 500 5 Mac} def
 /ucaron {(\165) (\365) 111 0 556 5 Mac} def
 /amacron {(\141) (\364) 83 0 500 5 Mac} def
 /emacron {(\145) (\364) 55 0 444 5 Mac} def
 /ecaron {(\145) (\365) 55 0 444 5 Mac} def
 /imacron {(\342) (\364) -27 0 333 5 Mac} def
 /icaron {(\342) (\365) -27 0 278 5 Mac} def
 /ocaron {(\157) (\365) 83 0 500 5 Mac} def
 /umacron {(\165) (\364) 111 0 556 5 Mac} def
 /ndieresis {(\156) (\367) 111 0 556 5 Mac} def
 /udiermacron {(\165) (\367) 111 0 (\364) 131 190 556 8 Mac} def
 /udieracute {(\165) (\367) 111 0 (\361) 131 190 556 8 Mac} def
 /udiercaron {(\165) (\367) 111 0 (\365) 131 190 556 8 Mac} def
 /udiergrave {(\165) (\367) 111 0 (\360) 131 190 556 8 Mac} def
end % Times-BoldItalic

/Helvetica CharCount 2 add dict def
Helvetica begin
 /acaron {(\141) (\365) 111 0 556 5 Mac} def
 /fulliacute {(\151) (\361) -27 225 278 5 Mac} def
 /omacron {(\157) (\364) 111 0 556 5 Mac} def
 /ucaron {(\165) (\365) 111 0 556 5 Mac} def
 /amacron {(\141) (\364) 111 0 556 5 Mac} def
 /emacron {(\145) (\364) 117 0 556 5 Mac} def
 /ecaron {(\145) (\365) 111 0 556 5 Mac} def
 /imacron {(\342) (\364) -27 0 333 5 Mac} def
 /icaron {(\342) (\365) -27 0 278 5 Mac} def
 /ocaron {(\157) (\365) 111 0 556 5 Mac} def
 /umacron {(\165) (\364) 111 0 556 5 Mac} def
 /ndieresis {(\156) (\367) 111 0 556 5 Mac} def
 /udiermacron {(\165) (\367) 111 0 (\364) 111 165 556 8 Mac} def
 /udieracute {(\165) (\367) 111 0 (\361) 111 185 556 8 Mac} def
 /udiercaron {(\165) (\367) 111 0 (\365) 111 185 556 8 Mac} def
 /udiergrave {(\165) (\367) 111 0 (\360) 111 185 556 8 Mac} def
end % Helvetica

/Helvetica-Oblique CharCount 2 add dict def
Helvetica-Oblique begin
 /acaron {(\141) (\365) 111 0 556 5 Mac} def
 /fulliacute {(\151) (\361) -27 225 278 5 Mac} def
 /omacron {(\157) (\364) 111 0 556 5 Mac} def
 /ucaron {(\165) (\365) 111 0 556 5 Mac} def
 /amacron {(\141) (\364) 111 0 556 5 Mac} def
 /emacron {(\145) (\364) 111 0 556 5 Mac} def
 /ecaron {(\145) (\365) 111 0 556 5 Mac} def
 /imacron {(\342) (\364) -27 0 333 5 Mac} def
```

```
  /icaron {(\342) (\365) -27 0 278 5 Mac} def
  /ocaron {(\157) (\365) 139 0 556 5 Mac} def
  /umacron {(\165) (\364) 111 0 556 5 Mac} def
  /ndieresis {(\156) (\367) 111 0 556 5 Mac} def
  /udiermacron {(\165) (\367) 111 0 (\364) 131 180 556 8 Mac} def
  /udieracute {(\165) (\367) 111 0 (\361) 131 180 556 8 Mac} def
  /udiercaron {(\165) (\367) 111 0 (\365) 131 180 556 8 Mac} def
  /udiergrave {(\165) (\367) 111 0 (\360) 131 180 556 8 Mac} def
 end % Helvetica-Oblique

 /Helvetica-Bold CharCount 2 add dict def
 Helvetica-Bold begin
  /acaron {(\141) (\365) 111 0 556 5 Mac} def
  /fulliacute {(\151) (\361) -27 225 278 5 Mac} def
  /omacron {(\157) (\364) 139 0 611 5 Mac} def
  /ucaron {(\165) (\365) 139 0 611 5 Mac} def
  /amacron {(\141) (\364) 111 0 556 5 Mac} def
  /emacron {(\145) (\364) 111 0 556 5 Mac} def
  /ecaron {(\145) (\365) 111 0 556 5 Mac} def
  /imacron {(\342) (\364) -27 0 333 5 Mac} def
  /icaron {(\342) (\365) -27 0 278 5 Mac} def
  /ocaron {(\157) (\365) 139 0 611 5 Mac} def
  /umacron {(\165) (\364) 139 0 611 5 Mac} def
  /ndieresis {(\156) (\367) 139 0 611 5 Mac} def
  /udiermacron {(\165) (\367) 139 0 (\364) 139 180 611 8 Mac} def
  /udieracute {(\165) (\367) 139 0 (\361) 139 200 611 8 Mac} def
  /udiercaron {(\165) (\367) 139 0 (\365) 139 190 611 8 Mac} def
  /udiergrave {(\165) (\367) 139 0 (\360) 139 200 611 8 Mac} def
 end % Helvetica-Bold

 /Helvetica-BoldOblique CharCount 2 add dict def
 Helvetica-BoldOblique begin
  /acaron {(\141) (\365) 111 0 556 5 Mac} def
  /fulliacute {(\151) (\361) -27 225 278 5 Mac} def
  /omacron {(\157) (\364) 139 0 611 5 Mac} def
  /ucaron {(\165) (\365) 139 0 611 5 Mac} def
  /amacron {(\141) (\364) 111 0 556 5 Mac} def
  /emacron {(\145) (\364) 111 0 556 5 Mac} def
  /ecaron {(\145) (\365) 111 0 556 5 Mac} def
  /imacron {(\342) (\364) -27 0 333 5 Mac} def
  /icaron {(\342) (\365) -27 0 278 5 Mac} def
  /ocaron {(\157) (\365) 139 0 611 5 Mac} def
  /umacron {(\165) (\364) 139 0 611 5 Mac} def
  /ndieresis {(\156) (\367) 139 0 611 5 Mac} def
  /udiermacron {(\165) (\367) 139 0 (\364) 159 180 611 8 Mac} def
  /udieracute {(\165) (\367) 139 0 (\361) 159 190 611 8 Mac} def
  /udiercaron {(\165) (\367) 139 0 (\365) 159 180 611 8 Mac} def
  /udiergrave {(\165) (\367) 139 0 (\360) 159 180 611 8 Mac} def
 end % Helvetica-BoldOblique
```

```
end % Accents

(%%[    Beginning MixFont Procedure                ]%%\n) print flush
/MixFont {
 /MixFD 25 dict def
 MixFD begin
  /CharInfo exch def /PlusFN exch def /ForFN exch def /BaseFN exch def
  /CharKnown CharInfo BaseFN known def
  BaseFN /Courier eq
  BaseFN /Courier-Bold eq and
  BaseFN /Courier-Oblique eq and
  BaseFN /Courier-BoldOblique eq and {/CharKnown false def} if
  /BaseFD BaseFN findfont def
  /ForFD 15 dict def
  /PlusFD BaseFD maxlength dict def
  BaseFD
   {exch dup /FID ne
    {dup /Encoding eq
      {exch dup length array copy PlusFD 3 1 roll put}
      {exch PlusFD 3 1 roll put}
     ifelse}
    {pop pop}
   ifelse
  } forall
  /OldCds CharInfo /PSCodes get def OldCds aload pop
  OldCds length 2 idiv {PlusFD /Encoding get 3 1 roll put} repeat
  CharKnown
   {PlusFD /FontName PlusFN put
    ForFD
     dup /FontName ForFN put
     dup /FontType 3 put
     dup /BaseFont PlusFN put
     dup /FontMatrix dup BaseFD exch get put
     dup /FontBBox dup BaseFD exch get put
         /Encoding 256 array put
    0 1 255 {ForFD /Encoding get exch /.notdef put} for
    /NewCds CharInfo /CompCodes get def NewCds aload pop
    NewCds length 2 idiv {PlusFD /Encoding get 3 1 roll put} repeat
    /NewProcs CharInfo BaseFN get def
    NewProcs dup
     /.notdef {(\040) stringwidth setcharwidth
               newpath 0 0 moveto (\040) show} bind put
     /.usual {charStr stringwidth setcharwidth
              newpath 0 0 moveto charStr show} bind put
    ForFD /CharProcs NewProcs put
    ForFD /BuildChar {MxDx /Combit get exec} put
    FontDirectory PlusFN known not {PlusFN PlusFD definefont pop} if
    FontDirectory ForFN known not {ForFN ForFD definefont pop} if}
```

```
    {(%%[  Font Re-Encoded - New Characters Unknown  ]%%\n) print flush
     PlusFD /FontName ForFN put
     FontDirectory ForFN known not {ForFN PlusFD definefont pop} if}
   ifelse
 end  % MixFD
} bind def
(%%[  MixFont procedures complete                  ]%%\n\n) print flush

/Courier /Courier-Foreign /C-Plus Accents MixFont
/Courier-Bold /Courier-Bold-Foreign /CB-Plus Accents MixFont
/Courier-Oblique /Courier-Oblique-Foreign /CO-Plus Accents MixFont
/Courier-BoldOblique /Courier-BoldOblique-Foreign /CBO-Plus Accents
      MixFont
/Helvetica /Helvetica-Foreign /H-Plus Accents MixFont
/Helvetica-Bold /Helvetica-Bold-Foreign /HB-Plus Accents MixFont
/Helvetica-Oblique /Helvetica-Oblique-Foreign /HO-Plus Accents MixFont
/Helvetica-BoldOblique /Helvetica-BoldOblique-Foreign /HBO-Plus Accents
      MixFont
/Times-Roman /Times-Roman-Foreign /TR-Plus Accents MixFont
/Times-Bold /Times-Bold-Foreign /TB-Plus Accents MixFont
/Times-Italic /Times-Italic-Foreign /TI-Plus Accents MixFont
/Times-BoldItalic /Times-BoldItalic-Foreign /TBI-Plus Accents MixFont
/AvantGarde-Book /AvantGarde-Book-Foreign /AGB-Plus Accents MixFont
/AvantGarde-Demi /AvantGarde-Demi-Foreign /AGD-Plus Accents MixFont
/AvantGarde-BookOblique /AvantGarde-BookOblique-Foreign /ABO-Plus
      Accents MixFont
/AvantGarde-DemiOblique /AvantGarde-DemiOblique-Foreign /ADO-Plus
      Accents MixFont
/Helvetica-Narrow /Helvetica-Narrow-Foreign /HN-Plus Accents MixFont
/Helvetica-Narrow-Bold /Helvetica-Narrow-Bold-Foreign /HNB-Plus Accents
      MixFont
/Helvetica-Narrow-Oblique /Helvetica-Narrow-Oblique-Foreign /HNO-Plus
      Accents MixFont
/Helvetica-Narrow-BoldOblique /Helvetica-Narrow-BoldOblique-Foreign
      /HNBO-Plus Accents MixFont
/Bookman-Light /Bookman-Light-Foreign /BL-Plus Accents MixFont
/Bookman-Demi /Bookman-Demi-Foreign /BD-Plus Accents MixFont
/Bookman-LightItalic /Bookman-LightItalic-Foreign /BLI-Plus Accents
      MixFont
/Bookman-DemiItalic /Bookman-DemiItalic-Foreign /BDI-Plus Accents
      MixFont
/NewCenturySchlbk-Roman /NewCenturySchlbk-Roman-Foreign /NSR-Plus
      Accents MixFont
/NewCenturySchlbk-Bold /NewCenturySchlbk-Bold-Foreign /NSB-Plus Accents
      MixFont
/NewCenturySchlbk-Italic /NewCenturySchlbk-Italic-Foreign /NSI-Plus
      Accents MixFont
/NewCenturySchlbk-BoldItalic /NewCenturySchlbk-BoldItalic-Foreign /NSBI-
      Plus Accents MixFont
```

```
/Palatino-Roman /Palatino-Roman-Foreign /PR-Plus Accents MixFont
/Palatino-Bold /Palatino-Bold-Foreign /PB-Plus Accents MixFont
/Palatino-Italic /Palatino-Italic-Foreign /PI-Plus Accents MixFont
/Palatino-BoldItalic /Palatino-BoldItalic-Foreign /PBI-Plus Accents
       MixFont
/ZapfChancery-MediumItalic /ZapfChancery-MediumItalic-Foreign /ZMI-Plus
       Accents MixFont
(\n%%[  LaserWriter preparation complete          ]%%\n\n) print flush
```

Bibliography

Adobe Systems, Inc., *AFM Files: An Interchange Format for PostScript Font Metrics*. Draft of June, 23 1986.

Adobe Systems, Inc., *PostScript Language Program Design*. Reading, MA: Addison-Wesley, 1988.

Adobe Systems, Inc., *PostScript Language Reference Manual*. Reading, MA: Addison-Wesley, 1987.

Adobe Systems, Inc., *PostScript Language Tutorial and Cookbook*. Reading, MA: Addison-Wesley, 1987.

Holzgang, David A., *Understanding PostScript Programming*. Alameda, CA: SYBEX, Inc., 1987.